ESD-TR-69-189
File Copy

ESLFE

THREE-DIMENSIONAL CURVES AND SURFACES FOR RAPID COMPUTER DISPLAY

AD# 696176

Theodore M. P. Lee

30 April 1969

ELECTRONIC SYSTEMS DIVISION

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Sponsored by: Advanced Research Projects Agency
Washington, D. C.

ARPA Order No. 952

(Prepared under Contract No. F19628-68-C-0379 by Harvard University, Cambridge, Massachusetts.)

AD0696176

ESD-TR-69-189

THREE-DIMENSIONAL CURVES AND SURFACES FOR RAPID COMPUTER DISPLAY

Theodore M. P. Lee

30 April 1969

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Sponsored by: Advanced Research Projects Agency
Washington, D. C.

ARPA Order No. 952

(Prepared under Contract No. F19628-68-C-0379 by Harvard University, Cambridge, Massachusetts.)

FOREWORD

This report describes work accomplished under Contract F-19628-68-C-0379 from June 1968 through April 1969. This contract is concerned with research on computer graphics and computer networking. In particular it is directed to development of new insights into the creation, analysis and presentation of information. This report presents an approach allowing three-dimensional information to become an element in the repertoire of the computing machine by analyzing a certain class of three-dimensional surfaces.

The report is based on a thesis submitted on April 30, 1969 by Theodore M. P. Lee to Harvard University, Division of Engineering and Applied Physics, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Professor Anthony G. Oettinger was the principal investigator for the contract. Dr. Lawrence Roberts was the ARPA director. Dr. Sylvia R. Mayer was the ARPA agent at Electronic Systems Division. Lt John P. McLean of Electronic Systems Division has provided technical guidance.

This technical report has been reviewed and is approved.

Sylvia R. Mayer
SYLVIA R. MAYER
Research Psychologist
Command Systems Division
Directorate of Planning and Technology

William F. Heisler
WILLIAM F. HEISLER, Colonel, USAF
Chief, Command Systems Division
Directorate of Planning and Technology

Lawrence G. Roberts
Lawrence G. Roberts
Special Assistant for Information Sciences
Advanced Research Projects Agency

ABSTRACT

Rational parametric polynomial functions of second degree or higher provide a class of curves including all conic sections. They can be generated by an iterative process easily implemented in software or hardware. The numerical accuracy of the process is analyzed. Algorithms for the specification, display, and modification of the curve are presented. Such curves are represented in a homogeneous coordinate formulation convenient for computer applications. Three-dimensional surfaces composed of such curves are similarly convenient to use. Without recourse to trigonometric functions such classical surfaces as spheres and toroids can be readily described. The ease with which translation, rotation and projective transformations can be applied is exhibited. In particular, we do not perform such transformations on the points of the surface to be displayed -- upwards of several thousand -- but rather upon the rather small set of numbers in a $4 \times 4 \times 4$ tensor that represents the surface. These surfaces are intended to be used in an interactive, freeform computer-aided design system. In this direction we discuss the enforcing of continuity conditions and possible data structures for representing the surfaces.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF FIGURES

LIST OF FIGURES (Concluded)

SECTION I

INTRODUCTION

1.1 GENERAL SUMMARY

This dissertation is concerned with the mathematics and some of the pragmatics of the specification, generation, and display of a certain class of three-dimensional surfaces and of the curves out of which the surfaces are formed. It is assumed that the reader has a good background in linear algebra and a fair background in the art of computer graphics. After providing a survey of the necessary material in the field of projective geometry we analyze the class of curves of interest followed by a detailed inspection of the surfaces. The final chapters present experimental results as well as a proposal for an evolving data structure relevant to the matter of surface specification.

Recent advances in computer display hardware have been concerned with achieving the ability to rapidly draw curves in two or three dimensions. Rational parametric polynomial functions of second degree or higher provide a class of curves including all conic sections. Such curves can be generated by an iterative process easily implemented in hardware or software. The mathematics for specifying, transforming and manipulating the curve are formulated in matrix algebra. In particular, transformations exist which alter the rate of display of the curve without changing its shape. Curves may be sectioned into smaller portions or expanded into larger continuations. The specific properties of rational parametric cubic polynomials are discussed in detail, including a statistical analysis of the propagation of truncation errors in their computation.

The thesis describes a class of three-dimensional surfaces -- formed by bi-cubic rational polynomials -- well-suited to computer display. The mathematical description of the surface is a homogeneous coordinate formulation of Coons' surfaces providing sufficient degrees of freedom to make them attractive for on-line shape design. Without recourse to trigonometric functions such classical surfaces as spheres and toroids are readily described.

The use of iterative computation techniques for generating the surface to provide rapid response is discussed. The properties of the boundary curves and interior contour curves (all rational parametric cubics) are examined. The ease with which translation, rotation and projective transformations can be applied is exhibited. In particular, we do not perform any such transformations on the points of the surface to be displayed -- upwards of several thousand -- but rather upon the rather small set of numbers in a tensor that represents the surface. Simple continuity constraints and other ways of specifying the surface from geometrical considerations are studied.

The above material -- as presented in Sections II and III -- is almost entirely original, although of course parallel work has been going on at a number of places. In particular, close contact has been maintained with S. A. Coons, so it is difficult to trace the origin of several ideas or the particular form of a derivation. Section I is entirely review and as such does not contribute to the state of the art.

In hoping to present this material for use in a viable interactive system, Appendix II proposes a data structure in which various intuitive as well as mathematical properties of the surfaces can be modelled. In applying some of the techniques of artificial intelligence projects as expressed in deductive systems it hopes to free the user from some of the restrictions of an algorithmic, pre-programmed environment. I claim originality for the particular use of these concepts and for the application of an associative data structure to the properties of surfaces, but must express the warning that much of this material is based on a cursory exploration of the possibilities and has no claim to finality.

The basic algorithms for the display of curves and surfaces have been programmed and are illustrated throughout the thesis. Most of the advanced theory -- such as specifying a surface from interior boundary curves -- has not been tested for lack of a suitably powerful and convenient interactive graphics facility with the capacity of three-dimensional input. The two to three second response time for the computation of a surface on the available equipment is just not adequate for any future experiments which would depend on the ability to obtain immediate feedback for the manipulation of a curve or surface in three dimensions.

Computer graphics -- the technology of composing and communicating two-dimensional, primarily non-textual, information with the aid of a computing machine [22] -- is new, but not new enough to let that be an excuse for inefficient techniques or imprecise formulation of ideas. This thesis proposes a mathematical formulation for a certain class of curves and surfaces in three dimensions. This formulation is suitable for efficient two-dimensional perspective display. The class of objects covered is sufficiently large as to be useful in the solution of a non-trivial set of the problems faced by people attempting to use computer graphics as an aid in the design of space forms. The class of surfaces, in yielding to direct mathematical characterization and analysis, is also interesting in itself. It is intended that this formulation be used in an interactive environment concerned with achieving rapid response in the realistic display of three-dimensional surfaces, without the large expenditure of computing resources needed in previously available approaches. The surfaces are either to be used as a rich vocabulary for some other problem or to be created directly as the objects of interest. Although the main goal of this work is the mathematical description of the surfaces and the presentation of the basic algorithms for their display, sufficient practical characterizations are presented so that this thesis may serve in some sense as a reference collection of directly applicable techniques. The mathematics is presented in a tensor notation easily transcribed to a working computer program. To aid the problem of interactively specifying a surface, characterizations have been developed which permit a small set of intuitively meaningful geometrical conditions -- such as tangent vector continuity -- to be used to describe the desired surface.

In summary, we have taken a simple concise idea, explored in detail its practical implementations, using any related theory as an aid to understanding, particularly through a set of alternative characterizations useful in different specific contexts.

Several existing techniques for modelling three-dimensional surfaces have used a coordinate dependent representation, such as $z = f(x, y)$ where f is of some predetermined type, say formed from polynomials in the two variables x and y. This approach may be satisfactory in the modelling of objects with well-defined dependent and independent axes, but it is completely contrived when applied to more general space forms, such as an automobile fender or bumper. For this reason, and for other reasons of historical accident, we have chosen a representation in which each coordinate is an independent function of two arguments, the two degrees of freedom on a surface. The functions have been chosen to be computationally efficient -- polynomials of degree three or less. Notice that the equations of some space curves lying on a surface can be obtained by treating one of the arguments as a constant and by letting the other be the parameter of the one degree of freedom on the curve.

Instead of the standard three-dimensional Cartesian representation of Euclidean space we have chosen to use a four-dimensional projective representation. This representation, in its abstract form, is taken from the realm of projective geometry in which it is the ratios among components rather than the components themselves that determine a point in space. In this four-dimensional space we represent a surface (strictly speaking, a part of a surface, called a patch) by four independent bi-cubic polynomial functions, one for each of the homogeneous coordinates. The parameters of the function range over the unit square, $0 \leqq u \leqq 1$, $0 \leqq v \leqq 1$. The four functions are represented by a 64 element tensor T where a component T_{ijk} is the coefficient of the term $u^i v^j$ in the polynomial for the k'th coordinate.

Such a homogeneous coordinate representation -- so named because abstractly four coordinates are used to represent three coordinates with no preferred mapping from the one space to the other -- has three particular advantages over the Cartesian representation for our work in computer graphics. Of primary importance, the use of four coordinates adds very convenient extra degrees of freedom allowing portions of such surfaces as spheres or toroids or of curves including all conic sections to be represented exactly with no more complicated functions than polynomials of second degree. In particular, the extra freedom provided by introducing homogeneous coordinates allows surfaces (and the curves of which they are composed) to be constrained with respect to tangent direction and magnitude at their boundaries, without sacrificing any internal structure. This freedom has not been possible in previous related work, such as cubic splines or Coons' bi-cubic surfaces [7].

Secondly, placing the abstract representation of a surface in homogeneous coordinates makes the application of the proper perspective projection strictly analogous to the application of the transformations of rotation and translation. With the particular form of a surface function chosen, we thus have a representation for objects independent of the manner in which they are to be viewed or composed into more complex objects. For instance, we display a visual representation of a surface by a set of parametrically orthogonal curves lying on the surface. The particular set used and the coarseness of the computer representation of the curves can be chosen independently of the abstract representation of the surface, as can the particular orientation and perspective viewpoint, all such choices being performed through simple matrix transformations of the abstract representation.

Lastly, the homogeneous nature of all the quantities (vectors, matrices, or tensors) used in a computer implementation of this work allows fixed point arithmetic to be used to a great speed advantage, with questions of scaling being relegated to the implicit scale factor present in all homogeneous arrays. For example, if the product of two matrices leads to arithmetic overflow, computation of the product is repeated with each element of one of the matrices being divided by successively larger powers of two until no overflow occurs; initially the matrices are separately normalized such that the largest magnitude of any of the numbers in each matrix is one.

This thesis is organized into three major sections -- Section II on curves, Section III on surfaces and Section IV discussing ways in which the results of the previous sections have been used in a practical implementation. The material on curves does not depend on the material on surfaces, while the latter uses explicitly only a few of the major results of the former. In both these sections the attempt has been to accompany the full formal presentation with a few special case examples to indicate the direction the exposition is taking. Throughout all the material there is a free alternation between theoretical consideration and explicit practical derivations. Wherever it seems appropriate, the mathematics has been presented in a more general form than would be strictly necessary for the practical derivations it accompanies.

Appendix II proposes a very general data structure particularly appropriate to the problems of representing these curves, surfaces, and the relations between them. Containing the least specific material in the thesis, it however is of importance in attempting to represent the types of interdependencies in the various topics, doing so without the necessary but confusing mathematical detail presented in Sections II and III. In some sense, it gives an answer to the question of what is the purpose of this research by presenting a framework in which it is meaningful for a designer to ask questions about the surfaces themselves, the answers to which can be found from the mathematical representations derived so tediously in the earlier sections.

To make this work moderately self-contained, there are a few subsections which are not strictly speaking original, but seem of such direct relevance as to merit inclusion. The subsection on homogeneous coordinates and transformations contains no new material, but serves to establish a common background for the rest of the thesis. The basic ideas for the iterative generation of curves are the joint efforts of S. A. Coons, D. Cohen, and the author; the backwards difference scheme was proposed by Coons in MAC-TR-41. The particular implementation discussed and the derivation of the necessary algebra and matrices to make it work are the author's responsibility, as is the error analysis. The subsection on surface construction utilizing interior curves was suggested to the author in considerable detail by Coons. To the best of my knowledge, the rest of the material is original. Most subsections of Sections II and III were presented in almost their present form as two separate papers to the 1969 Spring Joint Computer Conference [5, 13].

The remainder of this introductory section presents the context in which this work has arisen and furnishes a uniform notation and format for the homogeneous coordinate geometry. The subsection on notation should in particular be held firmly in mind when reading the mathematics of the surface formulation.

1.2 ANTECEDENTS AND HISTORICAL RELATIONS

The existence and direction of this research have been influenced by many factors, both intentional and accidental. The organization of this thesis is approximately in chronological sequence, but only because the topical dependencies guided the research in the same direction as seems best for presenting the material. In this brief subsection we consider the place of this work in its historical context.

In the initial stages the research was concerned purely with the problems of using the proposed Harvard Three-Dimensional Display [21] to draw curves as well as straight-line figures. Taking our direction from the earlier work of Blatt and Roberts at the M. I. T. Lincoln Laboratory [4, 16] using rational quadratics it was suggested that since the equipment was to use a 4×4 matrix we ought to consider adding a cubic term for additional flexibility. The results presented in Section II on curves indicated that this was indeed a promising approach and led to the inclusion of appropriate logic in the hardware to implement the forward difference scheme in the display.

After having in the course of investigating the properties of the curve formulation come to some basic understanding of the simplicity of the parametric polynomial approach, we suggested expanding the polynomial to two variables, thereby generating surfaces. Having demonstrated by construction the generation of the sphere and the cylinder, we abandoned immediate research on curves to proceed more directly with surfaces. Some time was spent in becoming familiar with Coons' earlier work [6, 7], but it was decided that as far as this research was concerned, adapting his more general formulation to homogeneous coordinates would lose the efficiency and simplicity of this approach. The notation has however been gratefully borrowed.

It soon became clear that the research would be progressing more rapidly than the construction of the three-dimensional display. We thus changed the character of the research slightly to make it more mathematical rather than directed to the utilization of this specific equipment. This change is what has limited experimental implementation to the rudimentary demonstration programs of Subsections 4.1 and 4.2 rather than to a more complete system, such as proposed in Appendix II. If we had proceeded in that direction, moreover, the resulting effort would probably not have been as general as is now proposed.

The evidence indicates that other work in this field -- either derived from Coons' formulation or from various implementations of interpolation in three dimensions -- as performed by the airframe or automobile industries is not strictly relevant to this formulation. Even though we have lost immediate contact with the three-dimensional hardware, we are concerned primarily with an efficient, flexible, relatively simple way of drawing three-dimensional curved surfaces subject to certain constraints. This is to enable an interactive approach to the specification of surfaces without requiring excessive computation power. The existing interactive systems of which we are aware are only approximately so, limited by ad hoc procedures relevant to the particular problem area (e.g., airframe specification and analysis) rather than to a more general class of problems encompassing a wide variety of surfaces. Strictly speaking, these results are not at all relevant to the problem of, for instance, fitting a smooth surface through a set of measured points on a clay model in order to drive a machine tool. These surfaces could be used for this kind of three-dimensional interpolation, but only rather clumsily.

1.3 NOTATION

We will be talking about surfaces, represented as tensors, curves, represented as matrices, or points, represented as either three-dimensional (ordinary coordinates) or four-dimensional (homogeneous coordinates) vectors. A vector, usually a point in homogeneous coordinates, will always be denoted by a vector arrow, for example, $\vec{V}$. Where relevant, a four-dimensional vector will be represented by an upper-case letter and a three-dimensional vector by a lower-case letter. The array representing a point or a curve may be independent or it may be a portion of a higher dimensional array representing a higher order object.

Subscripts will be used to denote <u>either</u> components of the array (tensor, matrix, or vector) <u>or</u> to indicate partial derivatives with respect to the parameters. Components of vectors will <u>not</u> have a vector arrow, although vector components of a matrix will, for example, the vector $\vec{A}_i$ of the matrix A_{ij}. Integer subscripts $_{i}$, $_{j}$, $_{k}$ or explicit numeric subscripts will be used for components in some cases while the symbols $_{hx}$, $_{hy}$, $_{hz}$, $_{h}$ will be used when it is desirable to emphasize the spatial coordinates. The subscripts $_{u}$, $_{v}$ will naturally refer to the partial derivatives $\frac{\partial}{\partial u}$, $\frac{\partial}{\partial v}$. The components of a point in three dimensions are indicated by subscripts $_{x}$, $_{y}$, $_{z}$ -- for example, p_x. This implies either true three-dimensional data or a division to remove the homogeneous scale factor.

When a tensor describing a surface is used <u>without</u> any subscripts for components it is to be treated not as an array of scalars but rather as a matrix whose elements are vectors. Multiplication of such a matrix of vectors by a matrix of scalars is to be interpreted in the standard way, with the summation being a vector sum and the individual multiplications being the product of a scalar and a vector.

When convenient, the standard convention of summing over repeated indices will be used, with the proviso that indices appearing on both sides of an equation will not be summed but indicate a running index. In some cases an indicial quantity will be used as an actual exponent -- context will make it clear when that occurs; in any case, if such an exponent is repeated in an equation it is to be treated according to the summation rule. Such summation and running indices will take on the values 0, 1, 2, 3 for virtually all of the thesis.

A curve $\vec{S}(u,v)$, u=a, a=constant, $0 \leqq v \leqq 1$, on a surface $\vec{S}(u,v)$ will be denoted by $\vec{S}^{u=a}$ or by $\vec{S}^{av}$ for brevity. A point, $\vec{S}(u,v)$, u=a, v=b, on the surface will be denoted by $\vec{S}^{u=a,\ v=b}$ or by $\vec{S}^{ab}$. In some cases these symbols will be further abbreviated by dropping the symbol S for the surface and writing just the values of the parameters -- see subsection 3.3. In most cases it does not matter whether such notation is taken to represent the object itself or the array describing the object; context should make the situation clear. The geometric and mathematical bases in which such an array represents an object will either be irrelevant or clearly specified.

1.4 HOMOGENEOUS COORDINATES

Taking the lead from the 19th century projective geometrists [1], computer display programmers have recently adopted the use of a <u>homogeneous coordinate</u> representation of geometrical data. This formulation is especially relevant when talking about perspective images. As is to be expected, most of the results derived by the mathematicians do not seem relevant to computer applications for they normally involve deductive rather than constructive proofs. This subsection summarizes a few of the definitions, conventions and formulae in homogeneous analytic projective geometry needed for an understanding of the surface formulation. Most of this material has been printed elsewhere, but is not readily available [14, 15, 17, 24].

Point: A point (in three dimensions) is represented by a non-zero vector of four components. Two points $\vec{P}$, $\vec{Q}$ are to be regarded as the same point if and only if they are linearly dependent; that is, if and only if there exists a non-zero constant α such that $\vec{P} = \alpha\vec{Q}$. The ordinary three-dimensional point $[x, y, z]$ can and will be represented in the homogeneous form as $[x, y, z, 1]$; hence any point $[a, b, c, d]$, $d \neq 0$, is equivalent to the three-dimensional point $[\frac{a}{d}, \frac{b}{d}, \frac{c}{d}]$. This representation will be indicated by the notation $[hx, hy, hz, h] = h\vec{v} = h[x, y, z, 1]$, $\vec{v} = [x, y, z, 1]$ for the homogeneous coordinates of a point $\vec{V}$, where by implication we take the quotients $\frac{hx}{h}, \frac{hy}{h}, \frac{hz}{h}$ to find the three-dimensional coordinates. The assumption is that once a point $[hx, hy, hz, h]$ has been obtained, something -- hardware or software -- will perform the necessary division by the homogeneous coordinate. In particular, whenever we talk about a point to be displayed, this division must be performed.

Line: Three points $\vec{P}$, $\vec{Q}$, $\vec{R}$ are collinear if and only if they are linearly dependent. The set of points $\vec{P}$ on the line through two points $\vec{Q}$, $\vec{R}$ can thus be generated parametrically by $\vec{P} = \alpha\vec{Q} + (1-\alpha)\vec{R}$, or, in full generality, by $\vec{P} = \alpha\vec{Q} + \beta\vec{R}$, α, β not both zero. Two points on this line, $\vec{P}_1 = \alpha_1\vec{Q} + \beta_1\vec{R}$ and $\vec{P}_2 = \alpha_2\vec{Q} + \beta_2\vec{R}$ are equivalent if and only if the vectors $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$ are linearly dependent, that is, in this case, proportional.

Plane: Four points $\vec{P}$, $\vec{Q}$, $\vec{R}$, $\vec{S}$, are coplanar if and only if they are linearly dependent. Hence the plane through three points $\vec{P}$, $\vec{Q}$, $\vec{R}$ is spanned by $\alpha\vec{P} + \beta\vec{Q} + \gamma\vec{R}$, α, β, γ not all zero. Furthermore, for the dependence $\alpha\vec{P} + \beta\vec{Q} + \gamma\vec{R} + \delta\vec{S} = 0$ to hold, we must have $|\vec{P}\ \vec{Q}\ \vec{R}\ \vec{S}| = 0$, or, expanding by minors on the fourth column, $\vec{S}\cdot\vec{T} = 0$ (vector dot product) where

$$\vec{T} = \left[\begin{vmatrix} P_2 & Q_2 & R_2 \\ P_3 & Q_3 & R_3 \\ P_4 & Q_4 & R_4 \end{vmatrix}, -\begin{vmatrix} P_1 & Q_1 & R_1 \\ P_3 & Q_3 & R_3 \\ P_4 & Q_4 & R_4 \end{vmatrix}, \begin{vmatrix} P_1 & Q_1 & R_1 \\ P_2 & Q_2 & R_2 \\ P_4 & Q_4 & R_4 \end{vmatrix}, -\begin{vmatrix} P_1 & Q_1 & R_1 \\ P_2 & Q_2 & R_2 \\ P_3 & Q_3 & R_3 \end{vmatrix} \right]$$

The relation $\vec{S}\cdot\vec{T} = 0$ is thus the equation of a plane and the vector $\vec{T}$ represents the plane. Among many results it can be shown that two planes $\vec{T}$ and $\vec{U}$ are equivalent if and only if the vectors $\vec{T}$ and $\vec{U}$ are proportional.

1.5 TRANSFORMATIONS

In ordinary three space, a non-singular matrix transformation of the form $Q_j = P_i T_{ij}$ (T a 3×3 matrix) is an affine transformation, producing only rotation and scaling. In the special three-dimensional space of homogeneous coordinates, such a transformation (where T is now 4×4) is called a projective transformation which in addition to rotation and scaling performs translation and a perspective transformation, hence its applicability to computer graphics. This transformation is derived as follows (see Figure 1.5.1 and Reference [22]):

Let an observer be at the origin $\vec{O}$ of a coordinate system and let a display screen S of size 2 scope units by 2 scope units be at position $z = \cot \alpha$, where α is half the angle subtended by the screen. The x coordinate x_s of the intersection with the screen of a ray from a point $\vec{P}$ to the observer -- perspective projection from $\vec{O}$ of $\vec{P}$ onto S -- is

$$x_s = \left(\frac{hx}{h}\right) \frac{1}{\left(\frac{hz}{h}\right)} \cot \alpha = \frac{hx}{hz} \cot \alpha$$

Similarly,

$$y_s = \frac{hy}{hz} \cot \alpha$$

and let us define an inverse distance as

$$z_s = \frac{h}{hz}$$

Then the point represented by

$$\vec{P}_s = \left[\frac{hx}{hz} \cot \alpha, \frac{hy}{hz} \cot \alpha, \frac{h}{hz}, 1\right]$$

gives as its x and y coordinates the location on the screen at which to draw the perspective projection of the point $\vec{P}$ and as its z coordinate a value monotonically related to distance, suitable for intensity modulation.

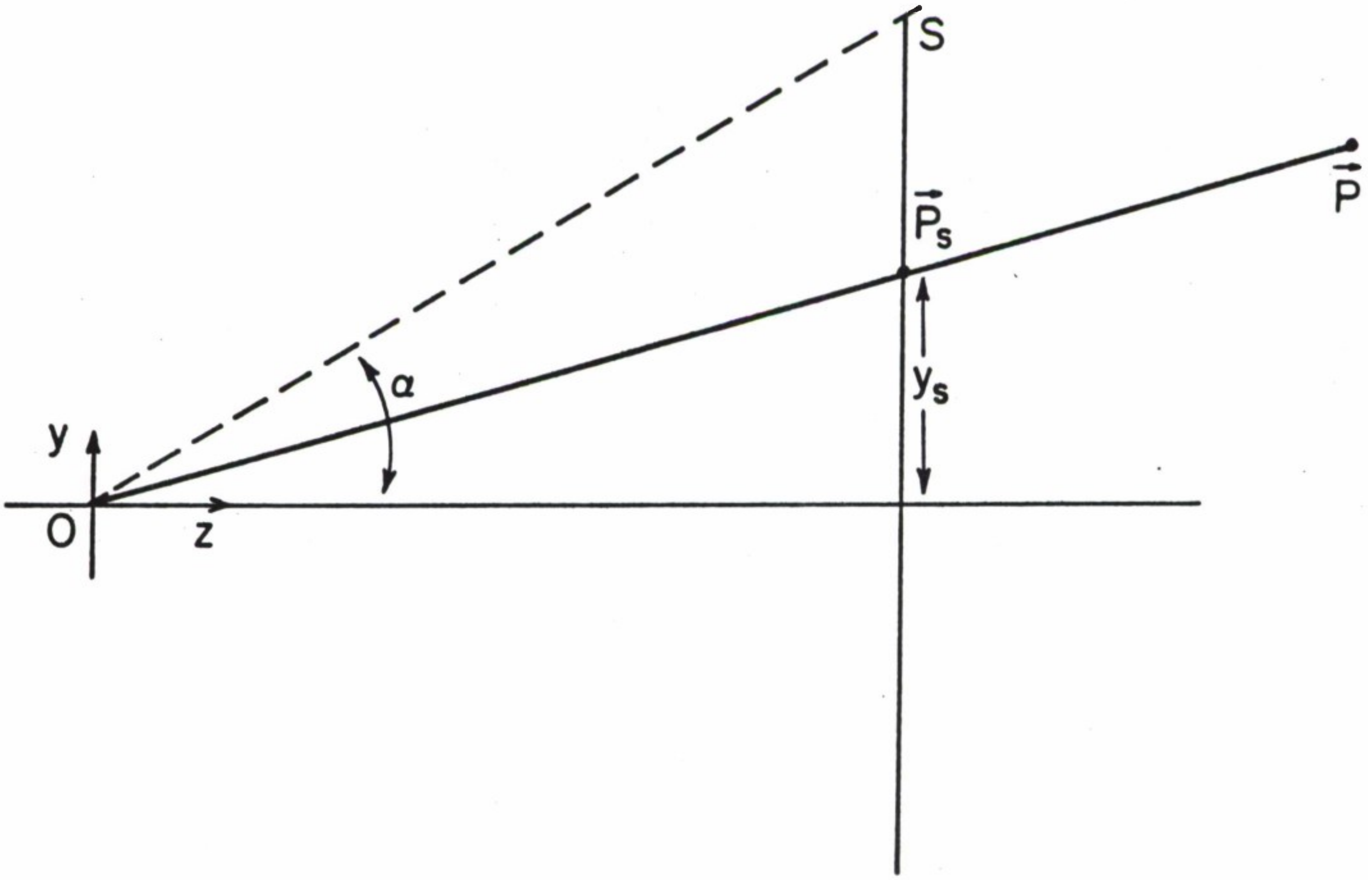


Fig. 1.5.1 Perspective Projection

$\vec{P}_s$ is equivalent to $(hz)\vec{P}_s = [hx \cot\alpha,\ hy \cot\alpha,\ h,\ hz] =$

$$= [hx, hy, hz, h] \begin{bmatrix} \cot\alpha & 0 & 0 & 0 \\ 0 & \cot\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and we have a matrix which performs a perspective projection on a point represented in homogeneous coordinates.

Translation is achieved by a matrix of the form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_t & y_t & z_t & 1 \end{bmatrix}$$

and rotation by a matrix of the form

$$\left[\begin{array}{ccc|c} & & & 0 \\ & R & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

where R is an ordinary three-dimensional rotation matrix. Typically, several such matrices -- rotation, translation, and projection -- will be applied in sequence to arrive at a single compound transformation expressed as the matrix product of the separate matrices.

Observe that given any matrix, M, which performs some transformation, a matrix αM, $\alpha \neq 0$, performs the same transformation, since $\alpha(\vec{P} \times M) = \vec{P} \times \alpha M$ implies $\vec{P} \times M$ and $\vec{P} \times \alpha M$ are proportional, for any point $\vec{P}$. In particular, the identity transformation becomes αI, $\alpha \neq 0$.

1.6 DISPLAY OF LINES

We have decided to use homogeneous coordinates for the representation of points in three-dimensional space with an intent to display such points in perspective on a two-dimensional screen. In general we will be talking about drawing straight lines rather than merely isolated points. Each such straight line will be represented by a starting point $\vec{P}$ and a finishing point $\vec{Q}$, both $\vec{P}$ and $\vec{Q}$ represented in homogeneous coordinates. To display a perspective view of a line we apply the perspective transformation to the end points of the line and display a two-dimensional vector between the projections of the endpoints.

Let $\vec{R} = \vec{P}\,T$ and $\vec{S} = \vec{Q}\,T$ where T is the desired perspective transformation matrix. Then $\left[\frac{R_1}{R_4}, \frac{R_2}{R_4}\right]$ and $\left[\frac{S_1}{S_4}, \frac{S_2}{S_4}\right]$ are the projected endpoints of the line on a display screen whose range is $-1 < x < 1$, $-1 < y < 1$. $\frac{R_3}{R_4}$ and $\frac{S_3}{S_4}$ are for the startpoint and endpoint, respectively, the reciprocal of the distance from the point to the observer, $\frac{h}{hz}$. If we apply this third coordinate to the intensification control on a display, we will obtain a picture in which the brightness varies approximately inversely with distance, depending on the response of the display to the intensification control. (The Adage commercial display uses a similar technique for depth indication.) As the vector generator drawing the line from $\vec{P}$ to $\vec{Q}$ moves the beam across the scope, we would want it to change the intensity in a correlated manner from the value at the start to the value at the finish. As $\frac{h}{hz} \rightarrow \infty$, $(z \rightarrow 0)$, we would have to limit the brightness to some fixed level; correspondingly we might not want $\frac{h}{hz} = 0$ $(z = \infty)$ to be represented as absolutely no intensity, but as some just barely visible minimum.

Notice the following two difficulties in a naive application of the above:

1. What if one or more of the ratios $\frac{R_1}{R_4}$, $\frac{R_2}{R_4}$, $\frac{S_1}{S_4}$, $\frac{S_2}{S_4}$ is greater than one in absolute value?

2. What if one or both of the intensities $\frac{R_3}{R_4}$, $\frac{S_3}{S_4}$ is less than zero? Condition 1 means that the point of the line causing the difficulty is outside the viewing window defined by the edges of the display. Condition 2 means that the appropriate point of the line is behind the observer ($z < 0$) and should not be visible.

The Clipping Divider [19] associated with the Harvard Three-Dimensional Display [21] was designed to solve the above problems by determining digitally before the division whether the endpoints of the line lie outside the viewing window, and, if they do, to compute the intersection of the line with the window in order that a vector may be drawn between two points within the display area. (See Figure 1.6.1.) The Clipping Divider does not compute the $\frac{h}{hz}$ term and does not deal with the problem of intensity modulation. It assumes that the homogeneous coordinate h of the original points is positive; it thus only uses the signs of R_4 or S_4 to tell whether the point is in front of or behind the observer and does not care how far. It does compute the intersection for a line which passes from in front of the observer to behind him, or vice versa.

In the software implementation of this clipping or windowing process, a modified version of a routine written by D. Cohen, we also do not provide for intensity modulation and avoid the $\frac{h}{hz}$ division. To include these would have slowed the program too much, and, would not have given a good result for the effort since the intensity control on the DEC type 340 display used is not suitable for the fine variation in intensity needed. Conversations with A. R. Forrest, Joint Computer-Aided Design Group, University of Cambridge, indicate that a half dozen or so intensity levels, appropriately spaced, selected by a software mapping from $\frac{h}{hz}$ gives a realistic appearance; even this level of intensity control was not readily possible with the equipment.

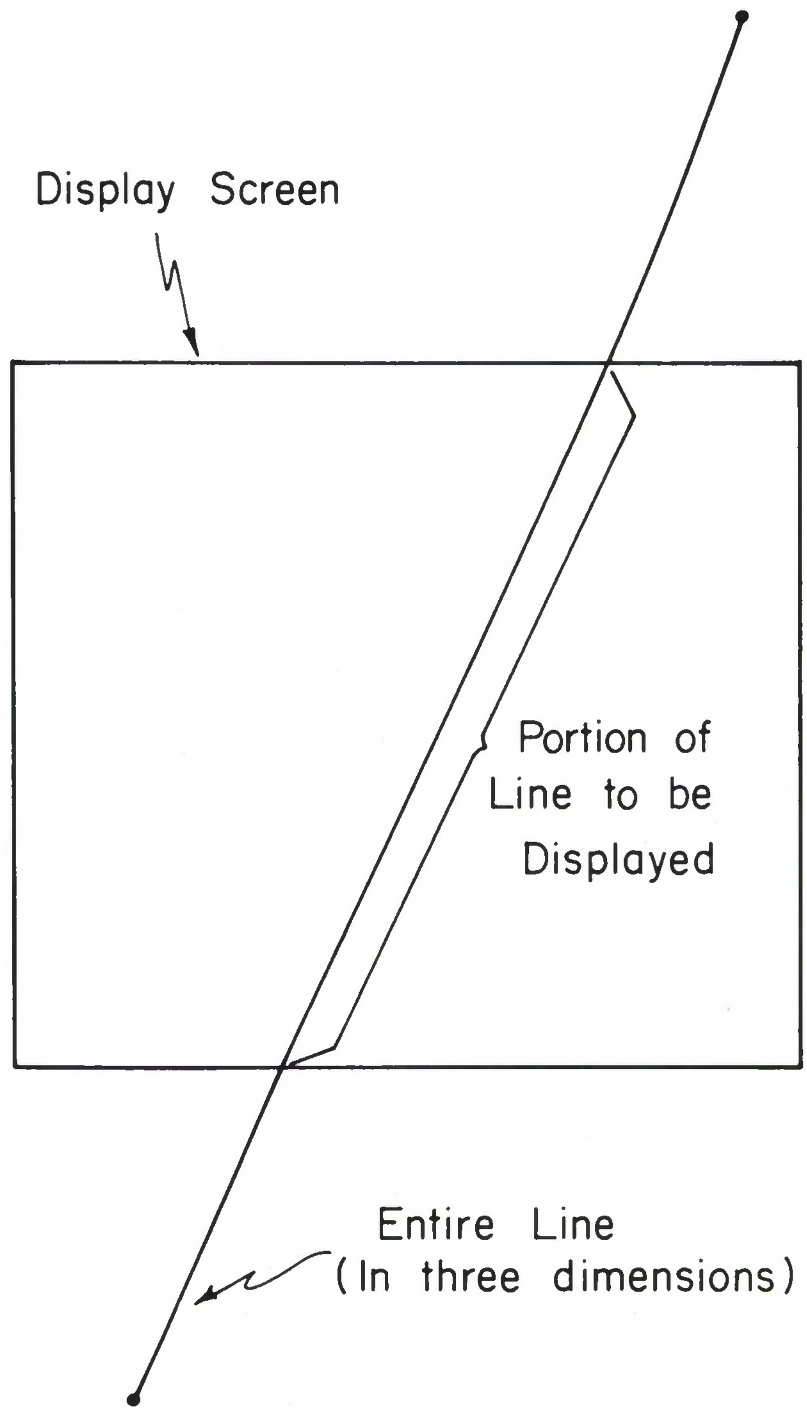

Display Screen
Portion of Line to be Displayed
Entire Line (In three dimensions)

FIG. 1.6.1 CLIPPING OF A LINE

SECTION II

CURVES

2.1 INTRODUCTION

Now that the computer displays with vector and character drawing capabilities are becoming a common form of on-line graphic output device, interest has turned toward providing a curvilinear display capability [4, 10, 16, 20]. Naturally, curves can be drawn as a series of short vectors stored in display memory, but the goal of current research is to provide hardware which draws a curve by generating such vectors, or perhaps, continuous beam motion, from a concise specification of a curve segment.

In this chapter we consider the properties of a particular class of curves suitable for such a hardware curve generator. (See Figures 2.1.1 and 2.1.2 for examples.) As it happens, the algorithms for generating successive points on the curve can be implemented efficiently in software.

Mathematically we present this material in as general a form as seems appropriate for the particular topic under discussion. Practically we think primarily of two-dimensional curves or three-dimensional curves projected into two dimensions.

We will consider the family of curve segments

$$\phi = \left\{ \vec{V} = \vec{V}(t), \quad 0 \leqq t \leqq 1 \right\}$$

where each component of $\vec{V}$ is a polynomial in t. We restrict the discussion to polynomials only because of the simplicity of polynomial evaluation. We initially consider these curves in R^N, the space of real vectors of dimension N; however, our interest lies in P^N, the perspective space of dimension N-1, generated from R^N by dividing each component of $\vec{V}$ by the last component.

Let ϕ_m be the family of curve segments defined by polynomials of degree not exceeding m. The higher m is, the more general is the

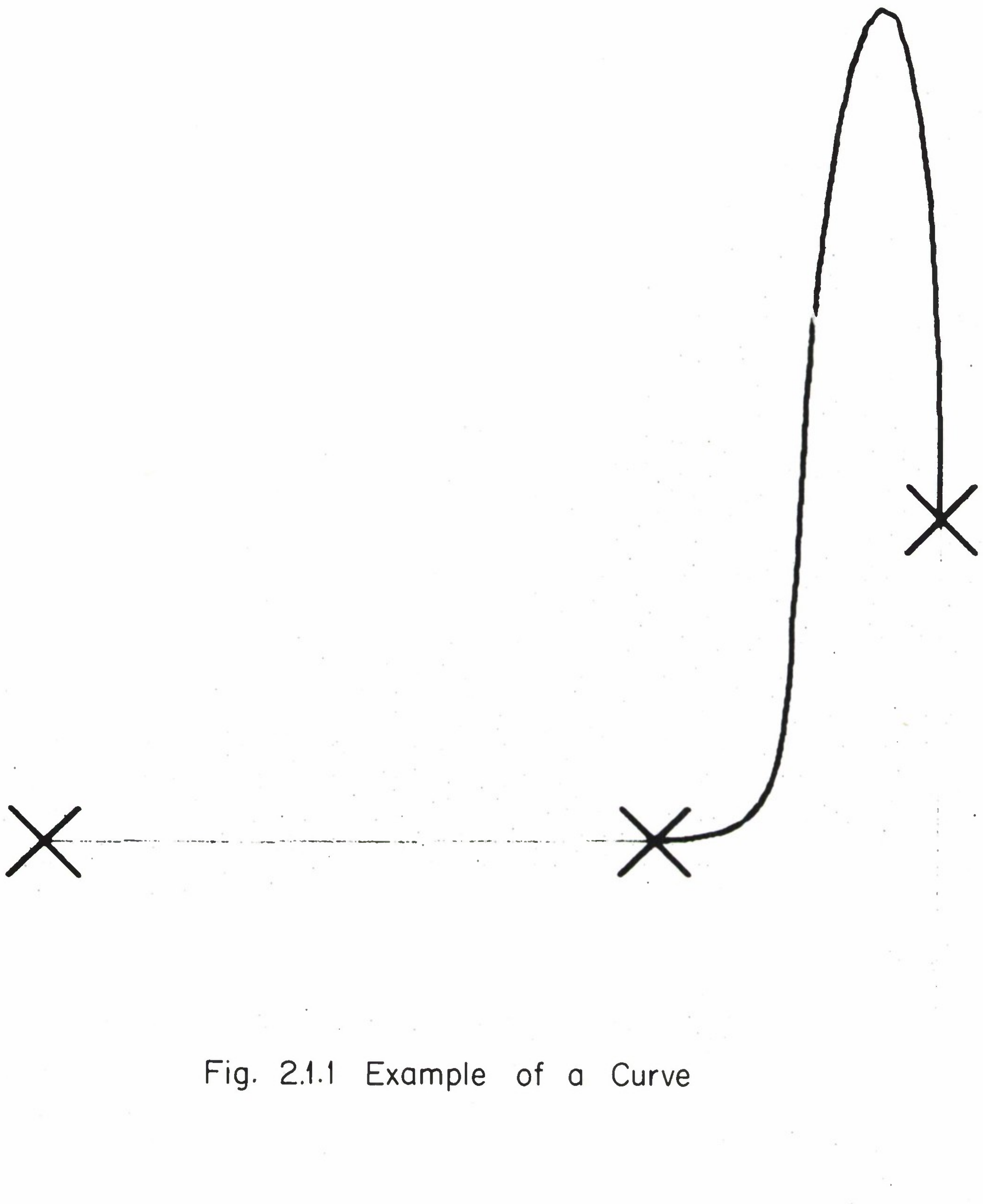


Fig. 2.1.1 Example of a Curve

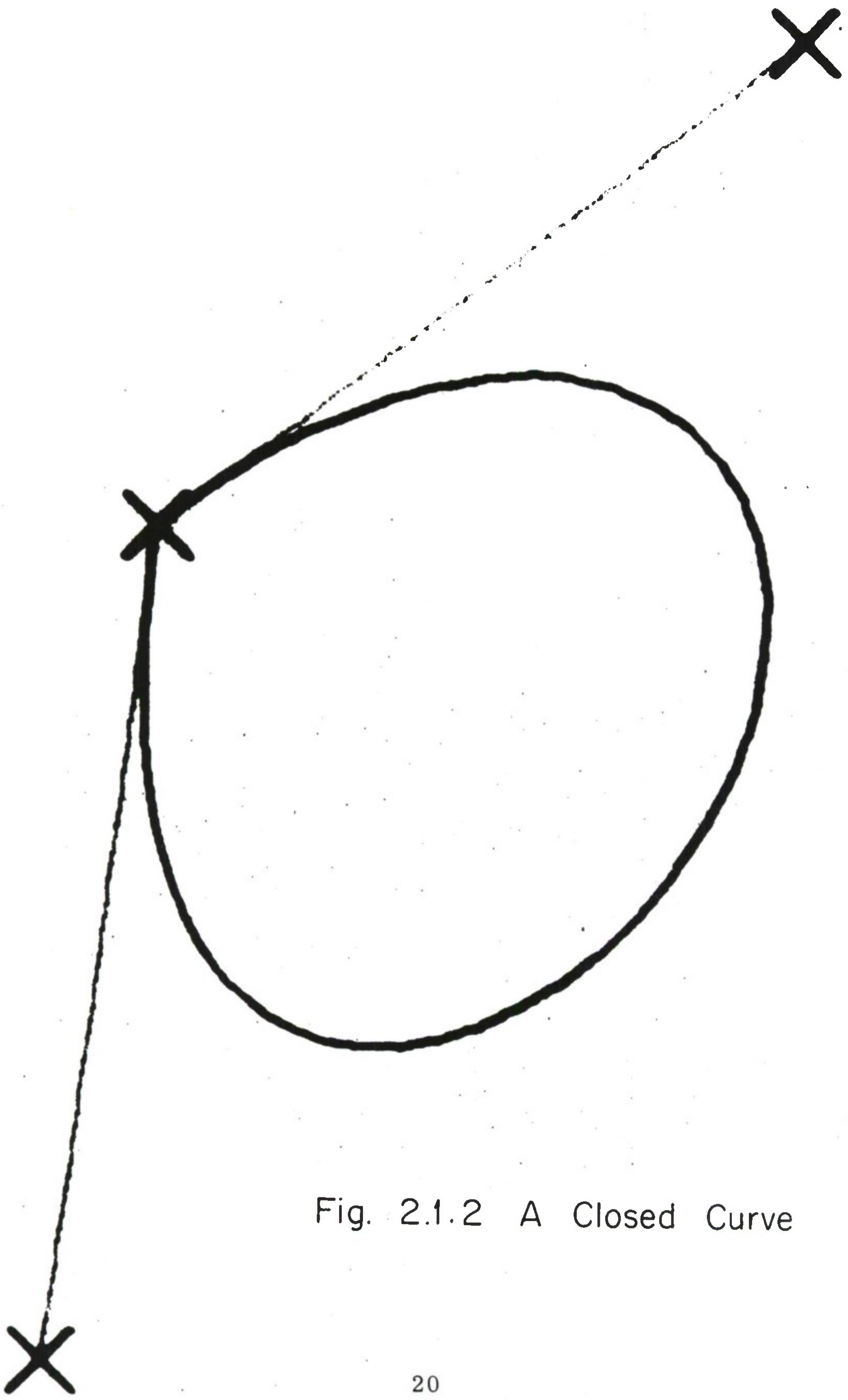


Fig. 2.1.2 A Closed Curve

family ϕ_m, but generating each point on any curve is more complex. The greater generality of ϕ_m for larger m enables the curve to satisfy more conditions, but correspondingly requires more conditions to uniquely define the curve. ϕ_2 includes straight lines and conic sections only. Therefore if $\vec{V} \varepsilon \phi_2$, $\vec{V}$ cannot cut itself and cannot have inflection points. ϕ_3 (which includes ϕ_2 as a proper subset) contains curve segments which may have inflection points and which may cut themselves, as shown in Figures 2.1.3 and 2.1.4.

2.2 ITERATIVE GENERATION OF CURVE

In order to display a curve we wish to compute successive points along the curve, forming a straight-line approximation to the curve by connecting adjacent points with short display vectors. Let

$$\vec{V}(t) = [V_1(t), V_2(t), V_3(t), V_4(t)]$$

be the equation of the curve in homogeneous coordinates. Each of the components $V_i(t)$ of $\vec{V}(t)$ is a polynomial in t. Let P(t) represent any of these polynomials. Although the successive points $\{t_1, t_2, t_3, \ldots, t_N\}$ could be any such that $0 \leqq t_0 < t_1 < \ldots < t_{N-1} < t_N \leqq 1$, we will choose to space the points equally along the parameter. Define $P_n = P(t_n) = P(n\delta)$ where $\delta = \frac{1}{N}$. We <u>could</u> compute the points by evaluating P for each t_n, but such an approach would be highly inefficient.

We choose therefore to use a finite difference method. Let ε be a scale factor (to be discussed shortly). Let f be any function. We define two difference operators with respect to the difference δ:

the forward difference: $\Delta f(x) = \frac{1}{\varepsilon}[f(x+\delta) - f(x)]$

the backward difference: $\nabla f(x) = \frac{1}{\varepsilon}[f(x) - f(x-\delta)]$

Δ^m and ∇^m are defined recursively where $\Delta^0 f = \nabla^0 f = f$. Note that $\frac{1}{\varepsilon}[P_{n+1} - P_n] = \Delta P_n = \nabla P_{n+1}$ and that if $P(t) = \sum_{k=0}^{m} a_k t^k$ then $\Delta^m P = \nabla^m P = \frac{1}{\varepsilon^m} m!\, a_m$ for any value of t. To be specific, we will

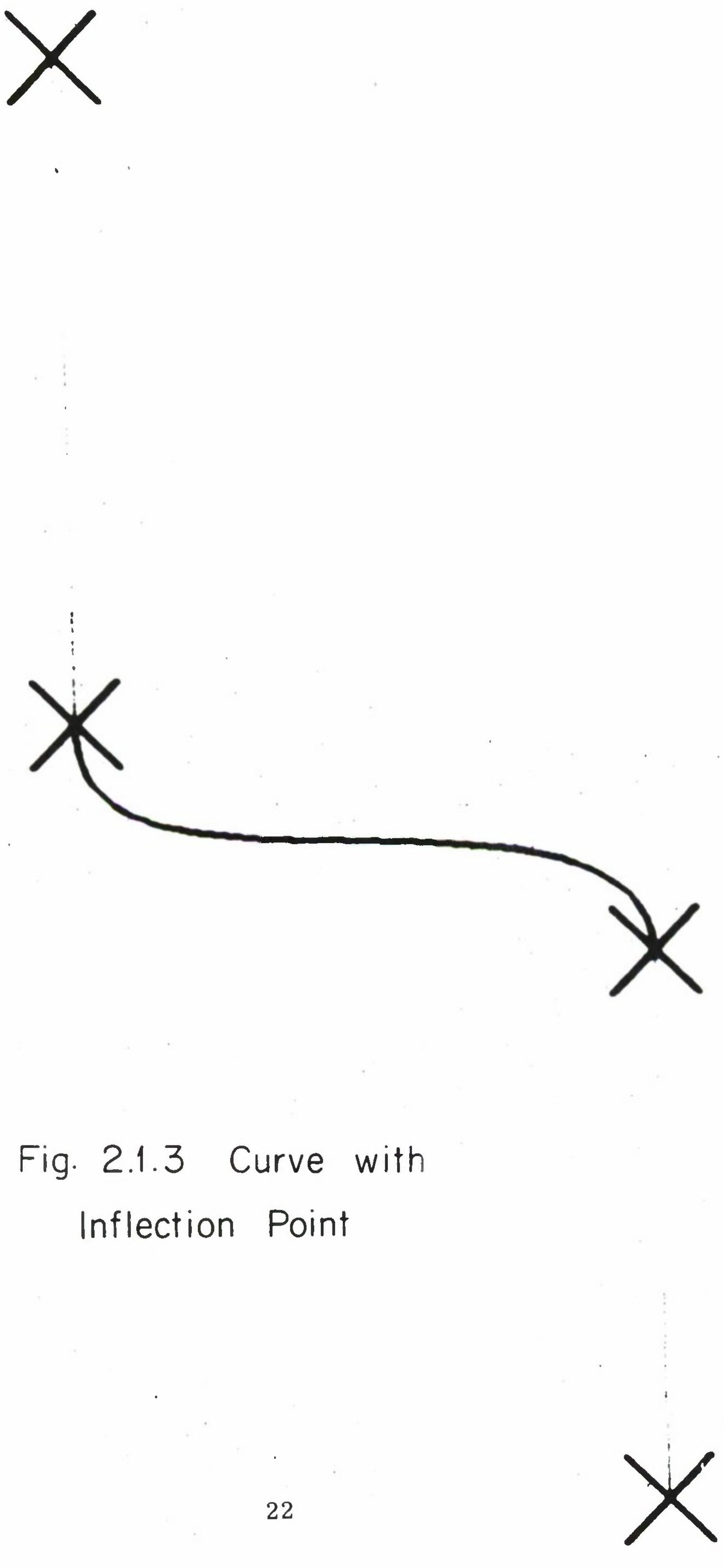


Fig. 2.1.3 Curve with Inflection Point

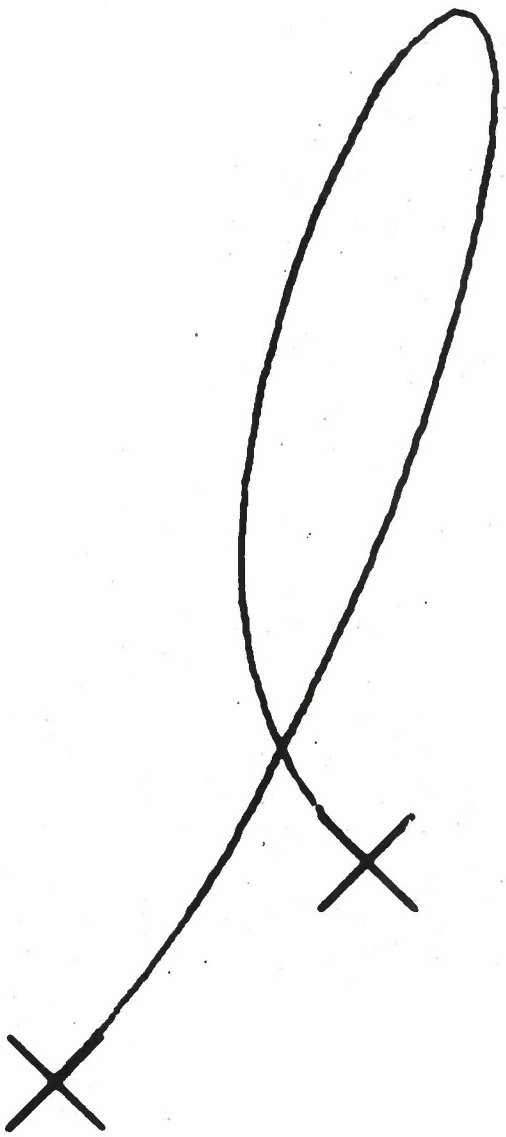


Fig. 2.1.4 Curve which Crosses Itself

choose to let m = 3, that is, we will talk about cubic polynomials.

If we define $P = [x^3 x^2 x 1]A$, $A \varepsilon R^4$, then

$$\begin{bmatrix} P(x) \\ \Delta P(x) \\ \Delta^2 P(x) \\ \Delta^3 P(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\varepsilon} & 0 & 0 \\ 0 & 0 & \frac{1}{\varepsilon^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\varepsilon^3} \end{bmatrix} \begin{bmatrix} x^3 & x^2 & x & 1 \\ 3x^2\delta + 3x\delta^2 + \delta^3 & 2x\delta + \delta^2 & \delta & 0 \\ 6x\delta^2 + 6\delta^3 & 2\delta^2 & 0 & 0 \\ 6\delta^3 & 0 & 0 & 0 \end{bmatrix} A$$

and

$$\begin{bmatrix} P(x) \\ \nabla P(x) \\ \nabla^2 P(x) \\ \nabla^3 P(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\varepsilon} & 0 & 0 \\ 0 & 0 & \frac{1}{\varepsilon^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\varepsilon^3} \end{bmatrix} \begin{bmatrix} x^3 & x^2 & x & 1 \\ 3x^2\delta - 3x\delta^2 + \delta^3 & 2x\delta - \delta^2 & \delta & 0 \\ 6x\delta^2 - 6\delta^2 & 2\delta^2 & 0 & 0 \\ 6\delta^3 & 0 & 0 & 0 \end{bmatrix} A$$

It is easy to see that

$$\begin{bmatrix} P_{n+1} \\ \Delta P_{n+1} \\ \Delta^2 P_{n+1} \\ \Delta^3 P_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon & 0 & 0 \\ 0 & 1 & \varepsilon & 0 \\ 0 & 0 & 1 & \varepsilon \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_n \\ \Delta P_n \\ \Delta^2 P_n \\ \Delta^3 P_n \end{bmatrix}$$

which we will abbreviate to $\vec{F}_{n+1} = S\vec{F}_n$ or as $\vec{F}_n = S^n\vec{F}_0$ and for the backward differences,

$$\begin{bmatrix} P_{n+1} \\ \nabla P_{n+1} \\ \nabla^2 P_{n+1} \\ \nabla^3 P_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon & \varepsilon^2 & \varepsilon^3 \\ 0 & 1 & \varepsilon & \varepsilon^2 \\ 0 & 0 & 1 & \varepsilon \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_n \\ \nabla P_n \\ \nabla^2 P_n \\ \nabla^3 P_n \end{bmatrix}$$

abbreviated as $\vec{B}_{n+1} = T\vec{B}_n$ or $\vec{B}_n = T^n\vec{B}_0$.

We can factor

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \varepsilon \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \varepsilon & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \varepsilon & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2A_1A_0$$

and

$$T = \begin{bmatrix} 1 & \varepsilon & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \varepsilon & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \varepsilon \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_0A_1A_2$$

where we identify A_i as the operation of scaling $\Delta^{i+1}P$ by ε and adding the scaled value to Δ^iP (or, $\nabla^{i+1}P$ to ∇^iP). These scaling and addition operations can be performed serially in the order A_0, A_1, A_2 for forward differences and in the opposite order A_2, A_1, A_0 for backward.

In both schemes there are only three additions; however, in the forward difference scheme, each new value Δ^iP_{n+1} depends on old values Δ^iP_n, $\Delta^{i+1}P_n$ whereas for the backward difference scheme each new value ∇^iP_{n+1} except for ∇^3P_{n+1} (which is constant) depends on both the old value ∇^iP_n and the new value of $\nabla^{i+1}P_{n+1}$. Thus all the operations of the forward difference scheme could be done in parallel, while those for the backward must be done serially. If the forward difference scheme is done in parallel, the old values Δ^iP_n cannot be destroyed until all the new values are obtained; if either scheme is done serially, each Δ^iP_n (or ∇^iP_n) can be replaced by Δ^iP_{n+1} (or ∇^iP_{n+1}) as soon as it is computed. Using serial programming, the backward difference scheme is slightly faster since the result ∇^iP_{n+1} of one operation is immediately available to compute the next value $\nabla^{i-1}P_{n+1}$ and need not be fetched from memory; in the forward scheme both the Δ^iP_{n-1} and $\Delta^{i+1}P_{n-1}$ needed to compute Δ^iP_n must be fetched.

In all the above we replace the polynomial P by the vector $\vec{V}$ and we have an efficient iterative method for generating the successive points $\vec{V}(n\delta)$ on the curve, keeping the 4 X 4 matrix of differences ($\Delta^i \vec{V}(n\delta)$ or $\nabla^i \vec{V}(n\delta)$) in memory or fast registers.

If we represent $\vec{V} = [t^3 t^2 t\, 1]A$, A a 4 X 4 matrix, then the initial set of differences is given by

$$\begin{bmatrix} \vec{V}(0) \\ \nabla \vec{V}(0) \\ \nabla^2 \vec{V}(0) \\ \nabla^3 \vec{V}(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{\delta^3}{\varepsilon} & \frac{-\delta^2}{\varepsilon} & \frac{\delta}{\varepsilon} & 0 \\ \frac{-6\delta^3}{\varepsilon^2} & \frac{2\delta^2}{\varepsilon^2} & 0 & 0 \\ \frac{6\delta^3}{\varepsilon^3} & 0 & 0 & 0 \end{bmatrix} A$$

or by

$$\begin{bmatrix} \vec{V}(0) \\ \Delta \vec{V}(0) \\ \Delta^2 \vec{V}(0) \\ \Delta^3 \vec{V}(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{\delta^3}{\varepsilon} & \frac{\delta^2}{\varepsilon} & \frac{\delta}{\varepsilon} & 0 \\ \frac{6\delta^3}{\varepsilon^2} & \frac{2\delta^2}{\varepsilon^2} & 0 & 0 \\ \frac{6\delta^3}{\varepsilon^3} & 0 & 0 & 0 \end{bmatrix} A$$

The scale factor ε is to be chosen to minimize the accumulated rounding error in computing $\vec{V}(t_n)$. We assume all numbers are stored as fixed point fractions. Since we are dealing in homogeneous coordinates, we can multiply

$$\begin{bmatrix} \vec{V}(0) \\ \Delta \vec{V}(0) \\ \Delta^2 \vec{V}(0) \\ \Delta^3 \vec{V}(0) \end{bmatrix}$$

by a normalization constant α in order to prevent any of the components of $\Delta^i \vec{V}(t_j)$ from exceeding one in magnitude. The normalization α should be chosen as large as possible in order that as much precision as possible be used. We will analyze the forward difference algorithm in terms of a single polynomial P rather than the vector $\vec{V}$, for simplicity.

We can write the forward difference scheme in matrix form as $\vec{F}_n = S^n \vec{F}_0 = (I+\varepsilon H)^n \vec{F}_0$ where in general $(H^k)_{ij} = \delta_i^{j+k}$ or for $k = 1, 2, 3$,

$$H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad H^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad H^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

and

$$H^{j>3} = 0$$

Expanding $(I+\varepsilon H)^n = \sum_i \binom{n}{i} \varepsilon^i H^i$, we get

$$\Delta^j P_n = \sum_{i=0}^{3-j} \binom{n}{i} \varepsilon^i \Delta^{j+i} P_0$$

For the backward difference scheme, we have similarly $\vec{B}_n = (I+\varepsilon H+\varepsilon^2 H^2+\varepsilon^3 H^3)^n \vec{B}_0$. Now we can write $(I+\varepsilon H+\varepsilon^2 H^2+\varepsilon^3 H^3) =$ $= (I-\varepsilon^4 H^4)(I-\varepsilon H)^{-1} = (I-\varepsilon H)^{-1}$ since $H^4 = 0$. Expanding formally, this gives $(I+\varepsilon H+\varepsilon^2 H^2+\varepsilon^3 H^3)^n = (I-\varepsilon H)^{-n} = \sum_{i=0}^{3} \binom{-n}{i} (-1)^i \varepsilon^i H^i$ where the summation stops at $i=3$ since $H^4 = 0$.

By a standard identity [12, p. 56], $\binom{-n}{i} = (-1)^i \binom{n+i-1}{i}$ and we have for the backwards differences,

$$\nabla^j P_n = \sum_{i=0}^{3-j} \binom{n+i-1}{i} \varepsilon^i \nabla^{j+i} P_0$$

We will analyze the forward difference scheme only, in order to determine optimal values for ε and δ. The analysis of the backwards scheme would be similar, but with additional algebraic complexity. Define $\overline{\Delta^j P_n} - \Delta^j P_n = e_n^j$ where $\Delta^j P_n$ is the correct value, $\overline{\Delta^j P_n}$ the computed value, and e_n^j the accumulated error. At any step of the process, we will have $\overline{\Delta^j P_n} = \rho_n^j + \overline{\Delta^j P_{n-1}} + \varepsilon \overline{\Delta^{j+1} P_{n-1}}$ where ρ_n^j is the roundoff error introduced in the computation. Define $e_0^j = \rho_0^j$, that is, the initial error in $\Delta^j P_0$ is due to roundoff.

Subtracting the original difference equation for $\Delta^j P_n$, we obtain $\overline{\Delta^j P_n} - \Delta^j P_n = \rho_n^j + (\overline{\Delta^j P_{n-1}} - \Delta^j P_{n-1}) + \varepsilon(\overline{\Delta^{j+1} P_{n-1}} - \Delta^{j+1} P_{n-1})$ or $e_n^j = \rho_n^j + e_{n-1}^j + \varepsilon e_{n-1}^{j+1}$.

If $\vec{E}_n$ denotes $\begin{bmatrix} e_n^0 \\ e_n^1 \\ e_n^2 \\ e_n^3 \end{bmatrix}$ and $\vec{R}_n = \begin{bmatrix} \rho_n^0 \\ \rho_n^1 \\ \rho_n^2 \\ \rho_n^3 \end{bmatrix}$ we have in vector form

$$\vec{E}_n = (I+\varepsilon H)\vec{E}_{n-1} + \vec{R}_n = (I+\varepsilon H)[(I+\varepsilon H)\vec{E}_{n-2} + \vec{R}_{n-1}] + \vec{R}_n =$$

$$= (I+\varepsilon H)^2 \vec{E}_{n-2} + (I+\varepsilon H)\vec{R}_{n-1} + \vec{R}_n =$$

$$= \sum_{i=0}^{n-1} (I+\varepsilon H)^i \vec{R}_{n-i} + (I+\varepsilon H)^n E_0 = \sum_{i=0}^{n} (I+\varepsilon H)^i \vec{R}_{n-i}$$

Expanding $(I+\varepsilon H)^i$ as before, we have $\vec{E}_n = \sum_{i=0}^{n} \sum_{k=0}^{3} \binom{i}{k} \varepsilon^k H^k \vec{R}_{n-i}$ or

$$e_n^j = \sum_{i=0}^{n} \sum_{k=0}^{3-j} \binom{i}{k} \varepsilon^k \left[\sum_{\ell} \delta_{\ell}^{j+k} \rho_{n-i}^{\ell} \right] = \sum_{i=0}^{n} \sum_{k=0}^{3-j} \binom{i}{k} \varepsilon^k \rho_{n-i}^{j+k} .$$

Now let us assume that the rounding errors are given by independent random variables with the same mean μ_0 and variance σ_0^2. We will also make use of the fact that $\Delta^3 P_n = \Delta^3 P_{n-1}$ $(\rho_n^3 = 0 \text{ for } n > 0)$, since no computation has to be done on the Δ^3 term. In other words,

$$\mu\rho_n^j = [1-\delta_3^j+\delta_3^j\delta_n^0]\mu_0 = [1-\delta_3^j(1-\delta_n^0)]\mu_0$$

and $$\sigma^2\rho_n^j = [1-\delta_3^j(1-\delta_n^0)]\sigma_0^2 .$$

Thus $$\mu e_n^j = \mu_0 \sum_{i=0}^{n} \sum_{k=0}^{3-j} \binom{i}{k} \varepsilon^k [1-\delta_3^{j+k}(1-\delta_{n-i}^0)] =$$

$$= \mu_0 \left[\sum_{i=0}^{n} \sum_{k=0}^{3-j} \binom{i}{k} \varepsilon^k - \sum_{i=0}^{n} (1-\delta_n^i)\binom{i}{3-j} \varepsilon^{3-j} \right] =$$

$$= \mu_0 \left[\sum_{i=0}^{n} \sum_{k=0}^{2-j} \binom{i}{k} \varepsilon^k + \binom{n}{3-j} \varepsilon^{3-j} \right]$$

By a standard identity [12, p. 54], $\sum_{i=0}^{n} \binom{i}{k} = \binom{n+1}{k+1}$, we get

$$\mu e_n^j = \mu_0 \left[\sum_{k=0}^{2-j} \binom{n+1}{k+1} \varepsilon^k + \binom{n}{3-j} \varepsilon^{3-j} \right]$$

To compute the variance $\sigma^2 e_n^j$ recall that for any sum of independent random variables,

$$\tilde{x} = \sum a_i x_i , \quad \sigma^2 \tilde{x} = \sum a_i^2 \sigma^2 x_i$$

so that

$$\sigma^2 e_n^j = \sigma_0^2 \left[\sum_{i=0}^{n} \sum_{k=0}^{2-j} \binom{i}{k}^2 \varepsilon^{2k} + \binom{n}{3-j}^2 \varepsilon^{2(3-j)} \right]$$

By a not-so-standard identity [11, p. 106],

$$\sum_{i=0}^{n} \binom{i}{k}^2 = \sum_{\ell=0}^{k} (-1)^\ell \binom{n+\ell+1}{k+\ell+1} \binom{n+1}{k-\ell}$$

and $$\sigma^2 e_n^j = \sigma_0^2\left[\sum_{k=0}^{2-j} \varepsilon^{2k} \sum_{\ell=0}^{k} (-1)^\ell \binom{n+\ell+1}{k+\ell+1}\binom{n+1}{k-\ell} + \binom{n}{3-j}^2 \varepsilon^{2(3-j)}\right]$$

The expanded expressions for the combinatorial sums are as follows:

$$\sum\binom{x}{0} = \binom{n+1}{1} = n+1$$

$$\sum\binom{x}{1} = \binom{n+1}{2} = \frac{(n+1)n}{2} = \frac{1}{2}(n^2-n)$$

$$\sum\binom{x}{2} = \binom{n+1}{3} = \frac{(n+1)n(n-1)}{6} = \frac{1}{6}(n^3-n)$$

$$\binom{n}{3} = \frac{n(n-1)(n-2)}{6} = \frac{1}{6}(n^3-3n^2+2n)$$

$$\sum\binom{x}{0}^2 = n+1$$

$$\sum\binom{x}{1}^2 = \frac{1}{3}n\left(n+\frac{1}{2}\right)(n+1) = \frac{1}{3}\left(n^3+\frac{3}{2}n^2+\frac{1}{2}n\right)$$

$$\sum\binom{x}{2}^2 = \frac{1}{20}\left(n+\sqrt{\frac{2}{3}}\right)(n+1)n(n-1)\left(n-\sqrt{\frac{2}{3}}\right) = \frac{1}{20}\left(n^5-\frac{5}{3}n^3+\frac{2}{3}n\right)$$

$$\binom{n}{3}^2 = \frac{n^2(n-1)^2(n-2)^2}{36} = \frac{1}{36}(n^6-6n^5+13n^4-12n^3+4n^2)$$

Notice that all the coefficients of the powers of n are of the same magnitude in each expression; we can thus reasonably approximate these error estimates by replacing each such expression by its first term, the approximation being asymptotically valid for $n \gg 1$. The resulting simplified expressions for the errors are thus

$$\mu e_n^0 = \mu_0\left[n+\frac{n^2}{2}\varepsilon+\frac{n^3}{6}\varepsilon^2+\frac{n^3}{6}\varepsilon^3\right] = \mu_0 n\left[1+\frac{n\varepsilon}{2}+\frac{(n\varepsilon)^2}{6}+\frac{(n\varepsilon)^3}{6n}\right]$$

$$\sigma^2 e_n = \sigma_0^2\left[n+\frac{n^3\varepsilon^2}{3}+\frac{n^5\varepsilon^4}{20}+\frac{n^6\varepsilon^6}{36}\right] = \sigma_0^2 n\left[1+\frac{(n\varepsilon)^2}{3}+\frac{(n\varepsilon)^4}{20}+\frac{(n\varepsilon)^6}{36n}\right]$$

For $n\varepsilon \to 0$, these expressions take on the limiting values $\mu e_n^0 = \mu_0 n$, $\sigma^2 e_n = \delta_0^2 n$ or $\sigma e_n = \sigma_0\sqrt{n}$.

The interpretation of these results is that most of the roundoff error occurs as introduced in the last stage ($V_n + \varepsilon\Delta V_n \rightarrow V_{n+1}$). A second conclusion is that it is significantly worth it to use an unbiased rounding scheme -- such as rounding on the basis of the first bit lost in the multiplication by ε -- rather than mere truncation. This is because the mean of the rounding errors propogates much faster than the standard deviation. In fact, if truncation errors permit only on the order of m steps to be taken, correct rounding will permit on the order of m^2 steps.

If the roundoff errors (μ_0', σ_0') in the initial $\Delta^j P_0$'s are larger than the errors occurring in the iterations (μ_0, σ_0) -- as would normally be the case since the $\Delta^j P_0$'s are the results of several computations -- the terms for

$$\sum_{k=0}^{3-j} \binom{n}{k} \varepsilon^k \rho_0^{j+k}$$

will have to be specifically added. This would not change the error significantly, since it would only add terms of the form $n^k \varepsilon^k \mu_0'$, whereas the corresponding terms already are of the form $n(n^k\varepsilon^k)\mu_0$, a factor of n larger, and similarly for σ.

The absolute error derived above is independent of the actual polynomial computed. The relative error, which measures the real accuracy of the computation, must be computed as $\frac{e_n^0}{P_n}$ and is dependent on $P(x)$. If we pick the normalization factor α as $\alpha = 1 / \left(\max_{k,j} |\Delta^j P_k| \right)$, the relative error is thus $\frac{e_n^0}{\alpha P_n}$. Figure 2.2.1 is a graph of the normalized relative mean error defined as $\frac{\mu e_n^0}{\alpha P_n \mu_0 n}$ for the polynomials $x^3 + x^2 + x + 1$, $x^2 + x + 1$ and $x + 1$, three typical cases. These graphs are for $n = 64$. The graphs for other n and for the normalized standard deviation of the error would be similar, both in shape and value.

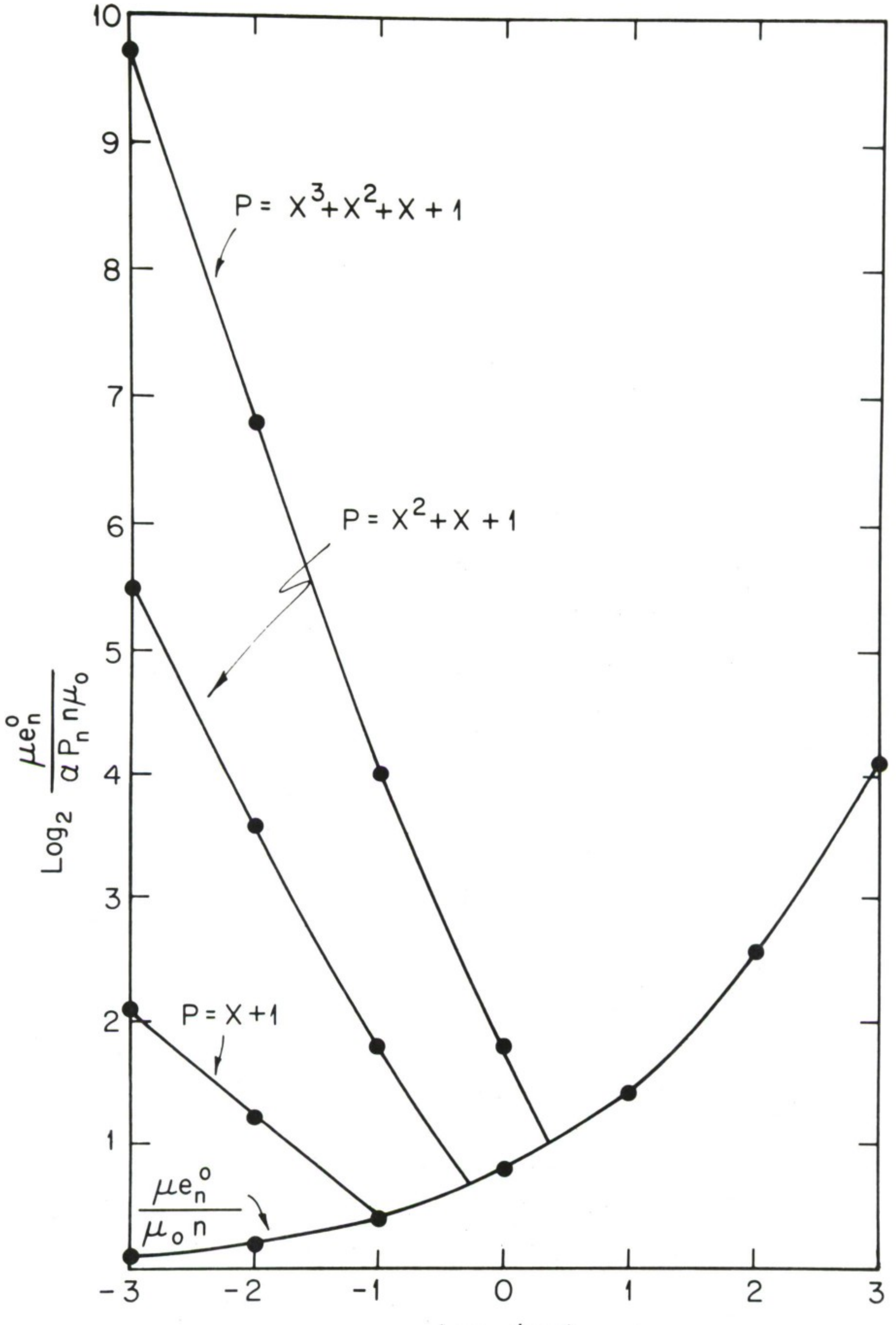


FIG. 2.2.1 NORMALIZED RELATIVE MEAN ERROR

For the case $P = x^3 + x^2 + x + 1$, the maximum occurs at $x = 1$, namely,

$$\begin{bmatrix} P_n \\ \Delta P_n \\ \Delta^2 P_n \\ \Delta^3 P_n \end{bmatrix} = \begin{bmatrix} 4 \\ \frac{6}{n\varepsilon} + \frac{4}{n \cdot n\varepsilon} + \frac{1}{n^2 \cdot n\varepsilon} \\ \frac{8}{(n\varepsilon)^2} + \frac{6}{n \cdot (n\varepsilon)^2} \\ \frac{6}{(n\varepsilon)^3} \end{bmatrix}$$

For $(n\varepsilon) < 1$, the dominating term is $\frac{6}{(n\varepsilon)^3}$ while for $(n\varepsilon) > 1$, the maximum becomes a constant 4. The relative errors are then given by dividing this maximum by the value assumed at P_n and multiplying by the absolute error. Using the asymptotic values for μe_n^0 and σe_n^0, we have

$$\left.\begin{array}{ll} \mu e_n^0 = \frac{6}{4(n\varepsilon)^3} \mu_0 n & \sigma e_n^0 = \frac{6}{4(n\varepsilon)^3} \sigma_0 \sqrt{n} \\ \mu e_n^0 = \frac{(n\varepsilon)^2}{6} \mu_0 n & \sigma e_n^0 = \frac{(n\varepsilon)^2}{\sqrt{20}} \sigma_0 \sqrt{n} \end{array}\right\} \begin{array}{l} 0 < n\varepsilon < 1 \\ \\ n \gg n\varepsilon \gg 1 \end{array}$$

The proper compromise is to choose $n\varepsilon = 1$, that is, $n = \frac{1}{\varepsilon}$ and $\delta = \varepsilon$, as confirmed by the exact data in Figure 2.2.1.

The final conclusions on selecting n and ε are thus:

1. Conclusion: Select n as small as possible consistent with the number of line segments needed to accurately represent the curve.

Reason: The mean of the error increases linearly with n and the standard deviation as the square root of n.

2. Conclusion: Select $\varepsilon = \frac{1}{n}$ and certainly not outside of the range $\frac{1}{2n} < \varepsilon < \frac{2}{n}$. ε may be chosen slightly less than optimal if the curve is relatively smooth.

Reason: The errors vary roughly with the square of $(n\varepsilon)$ or $\frac{1}{n\varepsilon}$, depending on which is worse, and, for $(n\varepsilon) < 1$ are worse the higher the degree of the polynomial.

Notice that the apparent error in generating higher order curves is particularly bad -- n must be large in order that the line segment approximation be good, and the presence of cubic terms heightens the error caused by scaling.

In applying this analysis to actual curve generation, we must recall that the scaling is applied to all components of $\vec{V}(t)$ equally; hence a cubic term in one increases the relative error in all.

Of course these are statistical estimates. However, if μ_0 is taken as the absolute value of the maximum rounding error, then μP_n will be an upper bound on the magnitude of the absolute accumulated error. On the other hand, assuming that the rounding errors are distributed randomly may not be valid -- certain polynomials with coefficients expressed exactly with few enough bits and drawn with few enough segments can be computed exactly. Experiments, however, do indicate that the relation of rounding error to $(n\varepsilon)$ is very strong, as predicted by theory.

These results were for generating curves. For drawing surfaces -- using a double iteration -- the absolute error is approximately twice as large and the scaling the square of the scaling in the case of curves. Thus for $P(x, y) = x^3y^3$, the relative error of the mean is on the order of $2 \cdot 36 \cdot n$, or, for $n = 32$, about 2^{11}, also verified by experiment.

A possible variation in this scheme would be to define

$$\Delta f(x) = \frac{1}{\varepsilon_1} (f(x+\delta) - f(x))$$

$$\Delta^2 f(x) = \frac{1}{\varepsilon_2} (\Delta f(x+\delta) - \Delta f(x))$$

$$\Delta^3 f(x) = \frac{1}{\varepsilon_3} (\Delta^2 f(x+\delta) - \Delta^2 f(x))$$

where the scaling factors ε_1, ε_2, ε_3 may be different for each difference. This method would permit the iteration to be more closely tailored to the specific polynomial being generated, but would require a more thorough and time-consuming analysis in order to select the proper values.

A further variation would be to make the difference δ different for each $\Delta^j f$, computing the more rapidly changing quantities f, Δf more frequently. This would be useful in the hardware implementation discussed next, as it would allow a relatively smooth movement of the beam between a set of coarsely spaced points t_i on the curve, the coarse points determined by Δ, Δ^2, and Δ^3, with f itself making a smoothly interpolated transition.

Roberts and Blatt [4, 16] have designed and built at the M. I. T. Lincoln Laboratory a curve generator using rational parametric quadratics. It uses a digital counter to generate a variable t, $0 < t < 1$, and then uses multiplying digital to analog converters to compute the polynomials, with a feedback loop to perform the homogeneous division. Figure 2.2.2 presents a suggestion for a curve generator using the iterative approach analyzed here. Conceptually, it would be somewhat better than Roberts' scheme, using only three (optionally four) multiplying D/A converters with only one converter in the feedback loop, whereas Roberts must include all the converters a, b, c making up a sum at^2+bt+c, as well as the converter generating t^2 and t.

The additions would be done at fixed clock intervals, either serially or in parallel, depending on your budget and desire for fast adders. The sign of the feedback amplifier would have to be chosen to make the loop stable, that is, opposite to the sign of hz. The third column is optional and gives an intensity signal for depth. Digital scaling by ε would be performed by having the lower inputs to each adder shifted, either by a fixed amount or selectable from a set of fixed amounts, say, two to six bits, meaning $\frac{1}{4} \leqq \varepsilon \leqq \frac{1}{64}$. Analog scaling would be done so that analog overflow would indicate either a point offscale on x or y or too close to the observer in z; such offscale portions of the curve would be blanked out.

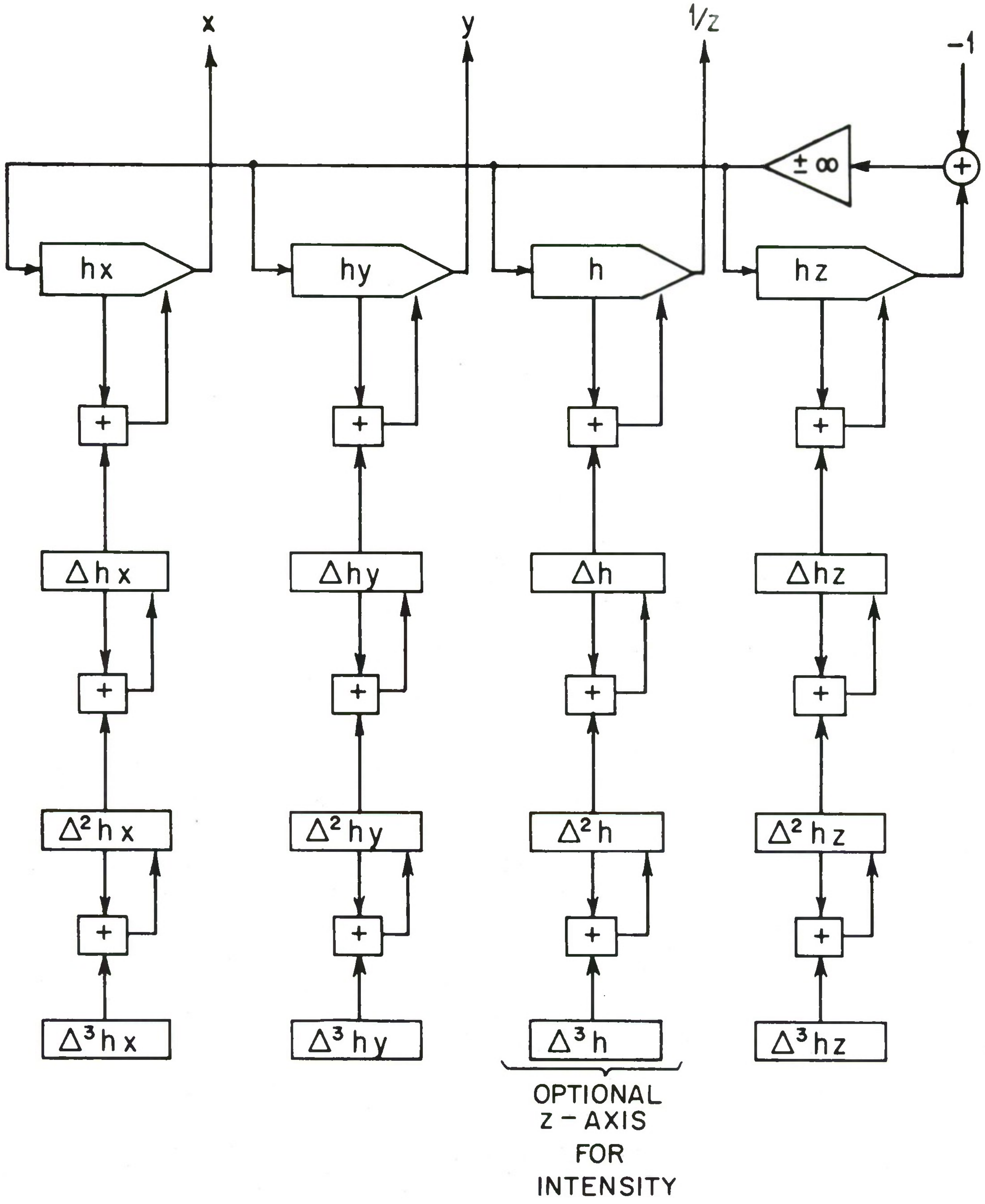


FIG. 2.2.2. CURVE GENERATOR USING MULTIPLYING D/A CONVERTERS.

2.3 BASIC CURVE FORM

For our purposes a curve in R^N is a vector-valued function from the real line R to the real vector space R^N of dimension N.* That is, a curve is an ordered N-tuple of real functions $(f_1, f_2, \ldots, f_N)$. Furthermore, let us restrict the functions f_i to be linear combinations of some linearly independent set of functions, $\phi = [\phi_0, \phi_1, \ldots, \phi_M]$ and let the functions be defined over the domain [0, 1]. Let $f_j(u) = \sum_{i=0}^{M} A_{ij}\phi_i(u)$ where the $\{A_{ij}\}$ are constants. Then a curve segment can be represented by a unique (M+1) $\times$ N real matrix A. A point $\vec{V}(u)$ on the curve associated with a parameter u is given by

$$\vec{V}(u) = [\phi_0(u), \phi_1(u), \ldots, \phi_M(u)]A = \vec{\phi}(u)A =$$

$$= [f_1(u), f_2(u), \ldots, f_N(u)], \quad 0 \leq u \leq 1.$$

In this context we will say either that the matrix A <u>represents</u> the curve, or, more pointedly, that the matrix A <u>is</u> the curve.

If we have some mapping $\vec{H}: R^N \to R^L$ from N dimensional into L dimensional space, then the image $\vec{v}(u) = \vec{H}(\vec{V}(u))$ of the curve $\vec{V}$ under the mapping will be a curve $\vec{v}(u)$ in R^L. The matrix A does not uniquely represent the curve $\vec{v}(u)$. We can define the equivalence class $\vec{H}[A]$ of a matrix A under the mapping as the set of all matrices A_H such that $\vec{H}(\vec{\phi}(u)A_H) = \vec{H}(\vec{\phi}(u)A)$; it is clear that all the curves in R^N specified by the $A_H \varepsilon \vec{H}[A]$ will be mapped into the same curve in R^L and there will thus be many representations of that curve. Notice that although the original function space -- $[f_1, f_2, \ldots, f_N]$ -- is a vector space of rank not exceeding (M+1) its image under the mapping $\vec{H}$ is not necessarily of finite rank and we cannot in general represent the resulting curve in R^L by an expression of the form $\vec{W}(u) = \vec{\Psi}(u)B$ where $\vec{W}(u) \varepsilon R^L$, $\vec{\Psi}(u) \varepsilon R^P$ and B a constant matrix of dimension P $\times$ L, $\vec{\Psi}(u)$ a fixed set of basis functions, dependent only on $\vec{H}$ and $\vec{\phi}$. The mapping $\vec{H}$

*In other words, we only consider curves in a parametric (explicit), rather than implicit, form.

will of course map the curve $[f_1, f_2, \ldots, f_N]$ into a curve $[g_1, g_2, \ldots, g_L]$ where $g_i = H_i[f_1, \ldots, f_N]$, where the $\{H_i\}$ are the components of the vector function $\vec{H}$.

We will usually choose the set of basis functions $[\phi_0, \ldots, \phi_M]$ to be the polynomials $[u^M, u^{M-1}, \ldots, u^2, u, 1]$ or some other set of linearly independent polynomials of degree M. The functions f_i will thus be the real polynomials of degree M or less. We will also choose N to be dimension 3 or 4 and L to be dimension 2 or 3 depending on whether we are talking about two or three dimensional curves. The mapping $\vec{H}$ will be that used in making the transformation from homogeneous coordinates to ordinary coordinates, namely, the projection obtained by dividing the first L components of the vector $\vec{V}$ (of dimension $L+1=N$) by the L+1st component to arrive at the L components of the vector $\vec{H}(\vec{V})$. We thus identify the curve specified by the form $\vec{\phi}A$ and the resulting vector $\vec{V}(u)$ as homogeneous coordinates of a curve $\vec{v}(u)$ in dimension one less. The many-to-one property of the mapping $\vec{H}$ is illustrated by the fact that two points in homogeneous coordinates, $\vec{P}$ and $\vec{Q}$, are equivalent if and only if $\vec{P} = \alpha\vec{Q}$ for some non-zero α.

Thus we talk about curves of the form

$$x = \frac{a_M u^M + a_{M-1} u^{M-1} + \ldots + a_1 u + a_0}{d_M u^M + d_{M-1} u^{M-1} + \ldots + d_1 u + d_0}$$

$$y = \frac{b_M u^M + b_{M-1} u^{M-1} + \ldots + b_1 u + b_0}{d_M u^M + d_{M-1} u^{M-1} + \ldots + d_1 u + d_0}$$

and optionally,

$$z = \frac{c_M u^M + c_{M-1} u^{M-1} + \ldots + c_1 u + c_0}{d_M u^M + d_{M-1} u^{M-1} + \ldots + d_1 u + d_0}$$

The value for each component of a point on the curve is the ratio of two polynomials in the parameter u. In theory and in practice we will first do some operations on the curve in R^N (or upon its representation as a matrix) and only at the last moment before display perform the mapping $H: R^N \to R^L$ by doing the division to remove the homogeneous coordinate. In most cases the many-to-one property of $\vec{H}$ will not come into play explicitly but only when we remember that the points $\vec{V}(u)$ are expressed in homogeneous coordinates.

If we have two sets of basis functions for the original function space $[f_1, \ldots, f_N]$, say $[\phi_0, \phi_1, \ldots, \phi_M]$ and $[\Psi_0, \Psi_1, \ldots, \Psi_M]$, we can always write the curve in two forms.

$\vec{V}(u) = [\phi_0, \phi_1, \ldots, \phi_M]A = [\Psi_0, \Psi_1, \ldots, \Psi_M]B$, $B = TA$ where T is the change of basis transformation, $[\Psi_0, \Psi_1, \ldots, \Psi_M]T = [\phi_0, \phi_1, \ldots, \phi_M]$. This relation will prove extremely useful in the practical matter of determining the matrix A to represent a curve under the basis $\vec{\phi} = [\phi_0, \phi_1, \ldots, \phi_M]$ where $\vec{\phi}$ will be chosen so as to make the display easy where a different basis $[\Psi_0, \ldots, \Psi_M]$ might be more convenient for specifying the curve or performing computations upon it.

2.4 REPARAMETERIZATION AND SHAPE INVARIANCE

Supposing we have a curve represented by the matrix A, it is natural to ask what matrices B represent the same curve. The major reason for investigating this shape invariance is to enable us to draw a particular curve in as smooth a manner as possible, while using equally spaced $\{u_i\}$. The initial choice of a matrix to represent a curve may be poorly made, as when the velocity vector $\frac{d\vec{v}}{du}$ is great at the same time the curvature of the curve is great.* We prefer to have the velocity slow at such points in order that successive points in time along the curve well represent its shape. (See Figure 2.4.1.)

We define two curves in R^L to be the same if and only if there exists a one-to-one correspondence between the points on the curves such that for each point $\vec{p}(s)$ on curve A there exists a point $\vec{q}(u)$ on

*We normally think of the parameter u as being time, and derivatives with respect to it as velocities, for reasons to become obvious later.

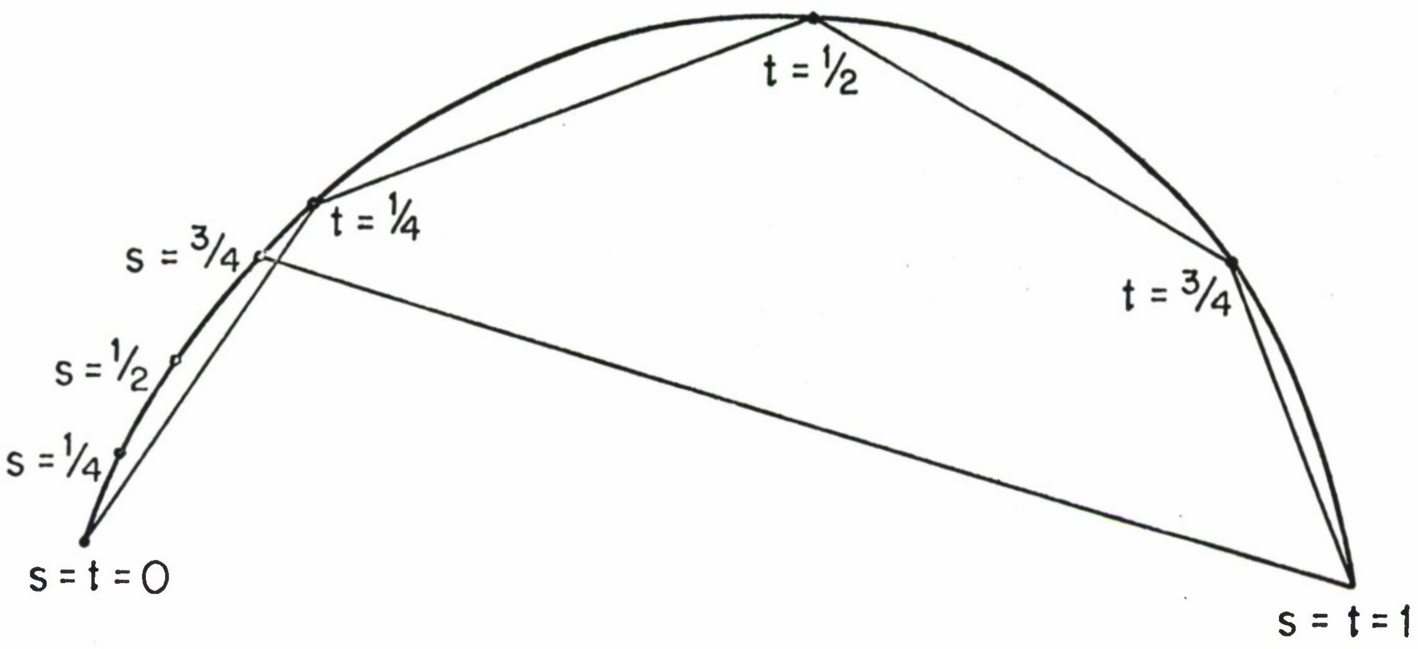


Fig. 2.4.1 Good and Bad Parameterizations

curve B such that $\vec{p}(s) = \vec{q}(u)$. This correspondence between the points on the curves induces a correspondence between the parameters on the curves, which we will denote by a function s such that $s = s(u)$, or $\vec{p}(s) = \vec{p}(s(u)) = \vec{q}(u)$. In this context we may have to extend the domain of one or the other of the curves to cover the entire real line, $-\infty < u < +\infty$, $-\infty < s < +\infty$, although we will still only display a finite portion, $\vec{V}(u) = \vec{\phi}(u)B$, and $0 \leqq u \leqq 1$. (See Figure 2.4.2.)

In the notation of the previous section, we have

$$\vec{H}\{[s^M \; s^{M-1} \; \ldots \; s \; 1]A\} = \vec{H}\{[u^M \; u^{M-1} \; \ldots \; u \; 1]B\}$$

where $\vec{H}$ is the operation of performing the homogeneous division. This equality between the coordinates on the curves will hold at the point $\vec{p}(s(u)) = \vec{q}(u)$ if and only if there exists a non-zero scalar factor α associated with the point $\vec{p}(s(u)) = \vec{q}(u)$ and hence a function $\alpha(u)$ of the parameter u such that

$$\alpha(u)\,[s^M \; s^{M-1} \; \ldots \; s \; 1]A = [u^M \; u^{M-1} \; \ldots \; u \; 1]B.$$

Write the above equation for M+1 different values of u, and get:

$$\begin{bmatrix} \alpha(u_0) & & & \\ & \alpha(u_1) & & \\ & & \ddots & \\ & & & \alpha(u_M) \end{bmatrix} \begin{bmatrix} s(u_0)^M & s(u_0)^{M-1} & \ldots & s(u_0) & 1 \\ s(u_1)^M & s(u_1)^{M-1} & \ldots & s(u_1) & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ s(u_M)^M & s(u_M)^{M-1} & & s(u_M) & 1 \end{bmatrix} A =$$

$$= \begin{bmatrix} u_0^M & u_0^{M-1} & \ldots & 1 \\ u_1^M & u_1^{M-1} & \ldots & 1 \\ \vdots & \vdots & & \vdots \\ u_M^M & u_M^{M-1} & & 1 \end{bmatrix} B$$

The matrix which precedes the B matrix is a Vandermonde matrix [2], which is non-singular since the $\{u_i\}$ are all different. Note that the

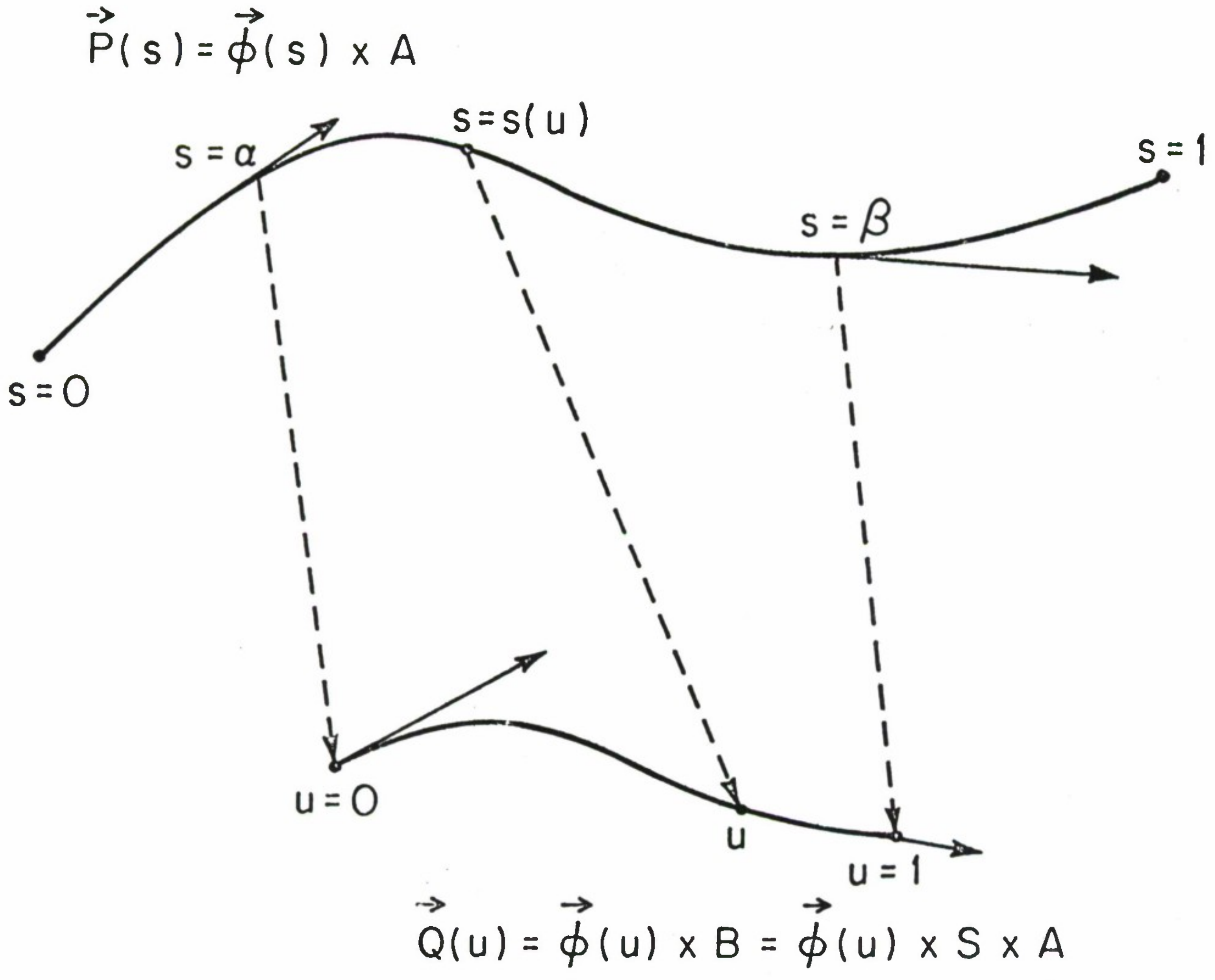

$\vec{P}(s) = \vec{\phi}(s) \times A$
$s = \alpha$
$s = s(u)$
$s = 1$
$s = \beta$
$s = 0$
$u = 0$
u
$u = 1$
$\vec{Q}(u) = \vec{\phi}(u) \times B = \vec{\phi}(u) \times S \times A$

Fig. 2.4.2 Shape Invariant Transformation

matrix preceding the A is also a non-singular Vandermonde matrix, as different $\{u_i\}$ imply different $\{s_i\}$.

Solve for B:

$$B = \begin{bmatrix} u_0^M & \cdots & 1 \\ \vdots & & \vdots \\ u_M^M & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} \alpha(u_0) & & \\ & \ddots & \\ & & \alpha(u_M) \end{bmatrix} \begin{bmatrix} s(u_0)^M & \cdots & 1 \\ \vdots & & \vdots \\ s(u_M)^M & \cdots & 1 \end{bmatrix} A$$

which we write as $B = SA$ for some as yet undetermined square $(M+1) \times (M+1)$ matrix S. Since S is the product of three non-singular matrices, it is non-singular.

The original equation becomes

$$\alpha(u)[s^M s^{M-1} \ldots 1]A = [u^M u^{M-1} \ldots 1]SA$$

which holds if:

$$\alpha(u)[s^M s^{M-1} \ldots 1] = [u^M u^{M-1} \ldots 1]S$$

If the rank of the null space of A is zero, then this will be a necessary condition as well. We will not pursue a further characterization of curves in terms of the null space of their associated matrix.

Under these conditions equating appropriate components of the vectors in the above equation gives:

(1) for the last $(M+1)^{st}$ component,

$$\alpha(u)\ 1 = [u^M \ldots 1]S \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = P(u)$$

P is a polynomial of degree not greater than M. If $P = \alpha(u)$ is a constant, then $s = u$ and $S = kI$ for some constant k, a trivial and uninteresting equality of curves. Hence we assume P is not a constant.

(2) for the Mth component:

$$\alpha(u)\, s = [u^M \ldots 1] S \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \\ 0 \end{bmatrix} = Q(u)$$

Q is a polynomial of degree not exceeding M and $s = \frac{Q}{\alpha(u)} = \frac{Q(u)}{P(u)}$.

(3) for the first component:

$$\alpha(u)\, s^M = [u^M \ldots 1] S \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = R(u)$$

R is a polynomial of degree not exceeding M and $s^M = \frac{R}{\alpha(u)} = \frac{R}{P} = \left[\frac{Q}{P}\right]^M$.

The expression $\frac{R}{P} = \left[\frac{Q}{P}\right]^M$ has at most M poles and at most M zeros (including multiplicities) since the polynomials R and P are of degree not greater than M. Thus $\frac{Q}{P}$ has at most one zero and one pole and we have

$$s = \frac{Q}{P} = \frac{au + b}{cu + d}\,.$$

Substituting in the original equation gives:

$$\alpha(u)\left[\left(\frac{au+b}{cu+d}\right)^M \left(\frac{au+b}{cu+d}\right)^{M-1} \ldots 1\right] = [u^M \ldots 1] S =$$

$$= \frac{\alpha(u)}{(cu+d)^M}\left[[(au+b)^M], [(au+b)^{M-1}(cu+d)], \ldots, [(cu+d)^M]\right]$$

Each component of $[u^M \ldots 1] S$ is a polynomial of degree not exceeding M in u. The components on the right-hand side above must thus be the same polynomials. Hence $\alpha(u) = e(cu+d)^M$, where e is a constant, and by inspection the component S_{ij} of the matrix S is the coefficient of u^{M-i} in the expansion of $e(au+b)^{M-j}(cu+d)^j$. Bypassing the algebra,

$$S_{ij} = e \sum_{k=0}^{j} a^{M-i-j+k}\, b^{i-k}\, c^{j-k}\, d^k \binom{M-j}{M-i-j+k}\binom{j}{k}$$

where we use the convention that $\binom{j}{k} = 0$ when j, k are outside of the range $0 \leqslant k \leqslant j$.

The matrix S for $M = 2$ is

$$\begin{bmatrix} a^2 & ac & c^2 \\ 2ab & bc + ad & 2cd \\ b^2 & bd & d^2 \end{bmatrix}$$

and the matrix for $M = 3$ is

$$\begin{bmatrix} a^3 & a^2c & ac^2 & c^3 \\ 3a^2b & 2abc + a^2d & bc^2 + 2acd & 3c^2d \\ 3ab^2 & b^2c + 2abd & 2bcd + ad^2 & 3cd^2 \\ b^3 & b^2d & bd^2 & d^3 \end{bmatrix}$$

Typically, we will want to use the reparameterization matrix S to change the amount (extent) of a given curve that is drawn and to alter the rate at which the moving vector $[u^M \; \ldots \; 1]SA$ traverses the curve. These variables can be specified by setting

1) $s(0) = \alpha$

2) $s(1) = \beta$

3) $\frac{s'(0)}{s'(1)} = r^2$

α is the value of the parameter s on the original curve A at which the new curve B begins. β is the value of the parameter s on A at which the curve B ends. r^2 is a ratio of derivatives; the significance of r will be shown shortly. It is easily verified that the ratio

$$\frac{s'(u_1)}{s'(u_2)} = \left[\frac{cu_2 + d}{cu_1 + d}\right]^2 \geq 0$$

Substituting the above conditions into $s = \frac{au + b}{cu + d}$, we solve for a, b, c, d:

$$\begin{aligned} a &= \beta - \alpha r \\ b &= \alpha r \\ c &= 1 - r \\ d &= r \end{aligned} \qquad \text{or} \qquad s = \frac{(\beta - \alpha r)u + \alpha r}{(1 - r)u + r}$$

using the arbitrary condition $c + d = 1$ to remove the extra degree of freedom introduced by the homogeneous form of the function s. The parameter e will allow an absolute adjustment of the homogeneous coordinate system, if that is desired.

The velocity vector $\frac{d\vec{v}}{ds}$ at the beginning ($u = 0$, $s = \alpha$) will be multiplied by the factor $\frac{\beta - \alpha}{r}$ and at the end ($u = 1$, $s = \beta$) by $(\beta-\alpha)r$. In fact, in general, we have

$$\frac{d\vec{p}}{du} = \frac{ds}{du}\frac{d\vec{p}}{ds} = \frac{(\beta-\alpha)r}{[(1-r)u+r]^2}\frac{d\vec{p}}{ds}$$

The product of the magnitudes of the velocity vectors at the ends of the new curve B is thus $(\beta-\alpha)^2$ times the product of the magnitudes of the velocity vectors at the corresponding points $s = \alpha$, $s = \beta$ of the original curve A. This product thus remains constant for all similar curves covering the same interval (α, β) of A. It is in any case independent of the rate parameter r. Notice that if $r < 0$, the new curve will not completely overlap the old curve in the range $0 \leq u \leq 1$, $0 \leq s \leq 1$ although over the full line $-\infty < u < +\infty$, $-\infty < s < +\infty$, the two curves will be identical. In particular, at the value $u = \frac{r}{r - 1}$, $s(u) = +\infty$, definitely outside the domain of the original curve $\vec{v}(s)$.

Thus, for example, if the matrix A draws half a circle, the matrix SA will draw the other half in the same direction if $\alpha = 1$, $\beta = 0$, $r = -1$. (See Figure 2.4.3 for an example of the use of this continuation matrix.)

For $M = 2$ this matrix $S(1, 0, -1)$ is

$$\begin{bmatrix} 1 & 2 & 4 \\ -2 & -3 & -4 \\ 1 & 1 & 1 \end{bmatrix}$$

and for $M = 3$ it is

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ -3 & -3 & -8 & -12 \\ 3 & 4 & 5 & 6 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

The matrix $S(1, 0, 1)$ would draw the same curve as A but in the opposite direction. This points out the important fact that the curves are directed, which must be taken into consideration when they are composed into surfaces, where the directions of opposite boundary curves makes a big difference on the surface between them.

If we wish only to change the parameterization of a curve without changing its extent, we have $\alpha = 0$, $\beta = 1$, and the reparameterization matrix for $M = 2$ is

$$S(r) = \begin{bmatrix} 1 & 1-r & (1-r)^2 \\ 0 & r & 2r(1-r) \\ 0 & 0 & r^2 \end{bmatrix}$$

and for $M = 3$

$$S(r) = \begin{bmatrix} 1 & (1-r) & (1-r)^2 & (1-r)^3 \\ 0 & r & 2(1-r)r & 3(1-r)^2 r \\ 0 & 0 & r^2 & 3(1-r)r^2 \\ 0 & 0 & 0 & r^3 \end{bmatrix}$$

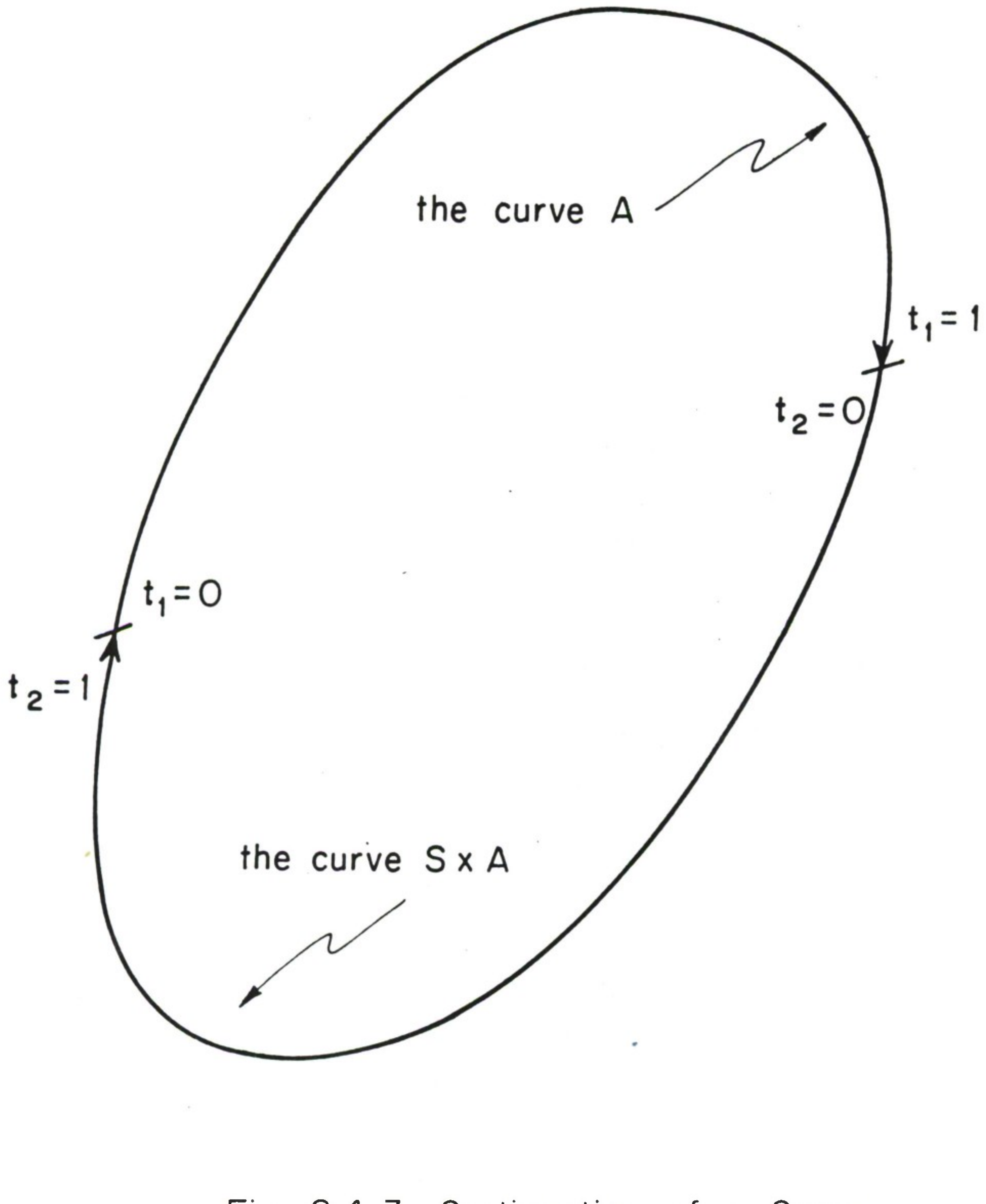


Fig. 2.4.3 Continuation of a Curve

This matrix can be used to adjust the homogeneous coordinate representation of a curve in an essentially arbitrary manner, as we now demonstrate. Let us assume a curve of the form

$$P(u) = [u^3\ u^2\ u\ 1]\begin{bmatrix} . & . & . & a \\ . & . & . & b \\ . & . & . & c \\ . & . & . & d \end{bmatrix} = [u^3\ u^2\ u\ 1]A$$

that is, $P_h(0) = d$, $P_h(1) = a + b + c + d$. Applying our matrix to the above, we obtain

$$Q(u) = [u^3\ u^2\ u\ 1]SA = [u^3\ u^2\ u\ 1]\begin{bmatrix} . & . & . & \alpha \\ . & . & . & \beta \\ . & . & . & \gamma \\ . & . & . & \delta \end{bmatrix}$$

where

$$\begin{aligned} \alpha &= a + b(1-r) + c(1-r)^2 + d(1-r)^3 \\ \beta &= \qquad b(r) + 2c(1-r)r + 3d(1-r)^2r \\ \gamma &= \qquad\qquad cr^2 + 3d(1-r)r^2 \\ \delta &= \qquad\qquad\qquad dr^3 \end{aligned}$$

and thus

$$Q_h(0) = \delta = dr^3 = r^3P_h(0)$$

$$Q_h(1) = \alpha + \beta + \gamma + \delta = a + b + c + d = P_h(1).$$

If we also apply a rescaling of the homogeneous coordinates by $R(u) = sQ(u)$, we have

$$R_h(0) = sr^3P_h(0)$$

$$R_h(1) = sP_h(1)$$

and we can make $R_h(0)$ and $R_h(1)$ completely arbitrary by setting

$$h(0) = R_h(0) = sr^3P_h(0)$$

$$h(1) = R_h(1) = sP_h(1)$$

and solving for

$$r = \left[\frac{h(0)\,P_h(1)}{h(1)\,P_h(0)}\right]^{1/3}$$

and

$$s = \frac{h(1)}{P_h(1)}$$

so long as none of the quantities are zero. In summary, given an arbitrary curve $P = [u^3\,u^2\,u\,1]A$, we obtain the same curve in the form $R = [u^3\,u^2\,u\,1]\,sS(r)A$ where we have completely specified the homogeneous coordinates as $R_h(0) = h(0)$, $R_h(1) = h(1)$. In other words, there are a doubly infinite number of matrix representations for a given curve segment.

If we have a curve $\vec{v}(u) = \vec{H}[\vec{\psi}A_T]$ represented by a matrix A_T for some other basis function $\vec{\psi} = [u^M\;u^{M-1}\;\ldots\;1]T$ then the reparameterization matrix S_T for this basis will be given by the similarity transformation $S_T = T^{-1}ST$ and the curve matrix after reparameterization will be $T^{-1}STA_T$.

2.5 ENDPOINT DERIVATIVE FORM

In the previous subsections we have been talking about curves of arbitrary degree; in this subsection we will be concerned only with curves of degree m=3. Let $h\vec{v} = \vec{V}$ denote the homogeneous coordinates of a point in R^N; that is, let $h = V_N$, $\vec{v} = \frac{\vec{V}}{V_N} = \frac{\vec{V}}{h}$ where $\vec{v}$ is a vector of the form $[v_1 v_2 \ldots v_{N-1}\;1]$. Then the equation of a curve is given parametrically by

$$\vec{V}(u) = h(u)\vec{v}(u) = \vec{\phi}(u)A$$

where $\vec{\phi}(u)$ are appropriate M-dimensional basis functions. The curve in $R^{N-1} = R^L$ is obtained by taking the first L components of the vector $\vec{v}(u)$; that is, by dividing out the homogeneous coordinate h(u) from the vector $h(u)\vec{v}(u)$ and dropping the last component.

Let

$$\vec{V}(0) = \vec{V}_0$$

$$\vec{V}(1) = \vec{V}_1$$

$$\vec{V}'(0) = \left.\frac{d\vec{V}(u)}{du}\right|_{u=0} = \vec{V}'_0$$

$$\vec{V}'(1) = \vec{V}'_1$$

then we have

$$\begin{bmatrix} \vec{V}_0 \\ \vec{V}_1 \\ \vec{V}'_0 \\ \vec{V}'_1 \end{bmatrix} = \begin{bmatrix} \vec{\phi}(0) \\ \vec{\phi}(1) \\ \vec{\phi}'(0) \\ \vec{\phi}'(1) \end{bmatrix} A$$

One can show that the square matrix

$$\begin{bmatrix} \vec{\phi}(0) \\ \vec{\phi}(1) \\ \vec{\phi}'(0) \\ \vec{\phi}'(1) \end{bmatrix}$$

is non-singular since the basis functions $\vec{\phi}(u)$ are by definition four linearly independent polynomials of degree 3. Define

$$M = \begin{bmatrix} \vec{\phi}(0) \\ \vec{\phi}(1) \\ \vec{\phi}'(0) \\ \vec{\phi}'(1) \end{bmatrix}^{-1}$$

and then

$$A = M \begin{bmatrix} V_0 \\ V_1 \\ V'_0 \\ V'_1 \end{bmatrix}$$

We now write

$$\vec{V}_0 = h_0\vec{v}_0$$
$$\vec{V}_1 = h_1\vec{v}_1$$
$$\vec{V}'_0 = \left.\frac{d(h\vec{v})}{du}\right|_{u=0} = h'_0\vec{v}_0 + h_0\vec{v}'_0$$
$$\vec{V}'_1 = h'_1\vec{v}_1 + h_1\vec{v}'_1$$

where $\vec{v}'_0$, $\vec{v}'_1$ are the derivatives with respect to the parameter u and where h_0, h_1, h'_0, h'_1 are meaningful only in homogeneous coordinates. Notice that the last component of $\vec{v}'_0$ and $\vec{v}'_1$ is now 0. Thus

$$A = M\begin{bmatrix} h_0 & 0 & 0 & 0 \\ 0 & h_1 & 0 & 0 \\ h'_0 & 0 & h_0 & 0 \\ 0 & h'_1 & 0 & h_1 \end{bmatrix}\begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix}$$

The curve is given by

$$\vec{\phi}(u)A = \vec{\phi}(u)M\begin{bmatrix} h_0 & & & \\ & h_1 & & \\ h'_0 & & h_0 & \\ & h'_1 & & h_1 \end{bmatrix}\begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix}$$

We can perform the multiplication $\vec{\phi}(u)M$ separately to define a new set of basis polynomials $[F_0 F_1 G_0 G_1] = \vec{\phi}M$ with transformation of basis matrix M itself.* We observe that

$$\begin{bmatrix} F_0(0) & F_1(0) & G_0(0) & G_1(0) \\ F_0(1) & F_1(1) & G_0(1) & G_1(1) \\ F'_0(0) & F'_1(0) & G'_0(0) & G'_1(0) \\ F'_0(1) & F'_1(1) & G'_0(1) & G'_1(1) \end{bmatrix} = \begin{bmatrix} \vec{\phi}(0) \\ \vec{\phi}(1) \\ \vec{\phi}'(0) \\ \vec{\phi}'(1) \end{bmatrix} M = M^{-1}M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*This notation is from Coons [7].

With these basis functions, a curve can always be represented by a matrix A_M of the form

$$A_M = \begin{bmatrix} h_0 & & & \\ & h_1 & & \\ h_0 & & h_0 & \\ & h_1 & & h_1 \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix}$$

since it is given by the equation

$$\vec{V}(u) = [F_0 F_1 G_0 G_1] \begin{bmatrix} h_0 & & & \\ & h_1 & & \\ h_0 & & h_0 & \\ & h_1 & & h_1 \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix}$$

We will sometimes write this endpoint-derivative form as

$$\vec{V}(u) = [F_0 F_1 G_0 G_1] \begin{bmatrix} (h\vec{v})_0 \\ (h\vec{v})_1 \\ (h\vec{v})'_0 \\ (h\vec{v})'_1 \end{bmatrix}$$

It will also prove useful to rewrite it in the form

$$\vec{V}(u) = [h_0 h_1 h'_0 h'_1] \begin{bmatrix} F_0 & & G_0 & \\ & F_1 & & G_1 \\ G_0 & & & \\ & G_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix}$$

or in standard form as

$$\vec{V}(u) = [(F_0h_0 + G_0h_0'), (F_1h_1 + G_1h_1'), (G_0h_0), (G_1h_1)] \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}_0' \\ \vec{v}_1' \end{bmatrix}$$

where now the matrix

$$A = \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}_0' \\ \vec{v}_1' \end{bmatrix}$$

is especially simple, although for a very special basis function constructed specifically for the particular set of homogeneous coordinates h_0, h_1, h_0', h_1'. This form shows that a point on the curve $\vec{V}(u)$ is always a linear combination of the vectors $\vec{v}_0, \vec{v}_1, \vec{v}_0', \vec{v}_1'$.

For these basis functions the reparameterization matrix becomes

$$S(r) = \begin{bmatrix} r^3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3(1-r)r^2 & 0 & r^2 & 0 \\ 0 & 3(1-r) & 0 & r \end{bmatrix}$$

using the similarity transform $M^{-1}SM$.

Notice that this result gives a particularly simple answer to the question of determining whether two matrices represent the same curve, namely, if in endpoint derivative form we have

$$\vec{B}(0) = \alpha r^3 \vec{A}(0)$$

$$\vec{B}(1) = \alpha \vec{A}(1)$$

$$\vec{B}'(0) = \alpha\, 3(1-r)r^2 \vec{A}(0) + \alpha r^2 \vec{A}'(0)$$

$$\vec{B}'(1) = \alpha\, 3(1-r) \vec{A}(1) + \alpha r \vec{A}'(1)$$

then and only then will the matrices A and B represent the same curve. α and r are arbitrary constants.

2.6 CURVE SPECIFICATION

In the previous subsection we have characterized a rational parametric cubic polynomial curve in terms of endpoints -- $\vec{v}_0$, $\vec{v}_1$ -- and tangent vectors at the endpoints -- $\vec{v}_0'$, $\vec{v}_1'$ -- with four remaining degrees of freedom, h_0, h_1, h_0', h_1'. In this subsection we investigate several ways of computing the numbers h_0, h_1, h_0', and h_1' in order to completely specify the curve from geometric consideration. (See Figure 2.6.1.) Although some or all of these results may be inappropriate for a particular application, the techniques used indicate the generality of the curve formulation and what we believe to be the proper way to attack the problem.

Let us require the curve to pass through the point $\vec{v}_c$ at the value u_c of the parameter. We have

$$\vec{v}_c = [h_0 h_1 h_0' h_1'] \begin{bmatrix} F_0(u_c) & 0 & G_0(u_c) & 0 \\ 0 & F_1(u_c) & 0 & G_1(u_c) \\ G_0(u_c) & 0 & 0 & 0 \\ 0 & G_1(u_c) & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}_0' \\ \vec{v}_1' \end{bmatrix} = \vec{h}FV$$

We can solve $\vec{v}_c = \vec{h}FV$ for $\vec{h}F$ (and hence for $\vec{h}$, since F is nonsingular for any $0 < u_c < 1$) if and only if $\vec{v}_c$ belongs to the range of V. In R^3 the geometrical meaning of this condition is that if the curve is planar, $\vec{v}_c$ must be in the same plane, or, if the curve is linear, $\vec{v}_c$ must be on the same line. If the curve is really three-dimensional, then V will have full rank and we can solve directly for

$$\vec{h} = \vec{v}_c V^{-1} F^{-1} = \vec{v}_c (FV)^{-1}$$

In R^2 the equation $\vec{v}_c = \vec{h}FV$ is indeterminate since we have three equations and four unknowns. If $\vec{v}_c$ is in the range of V, we can impose

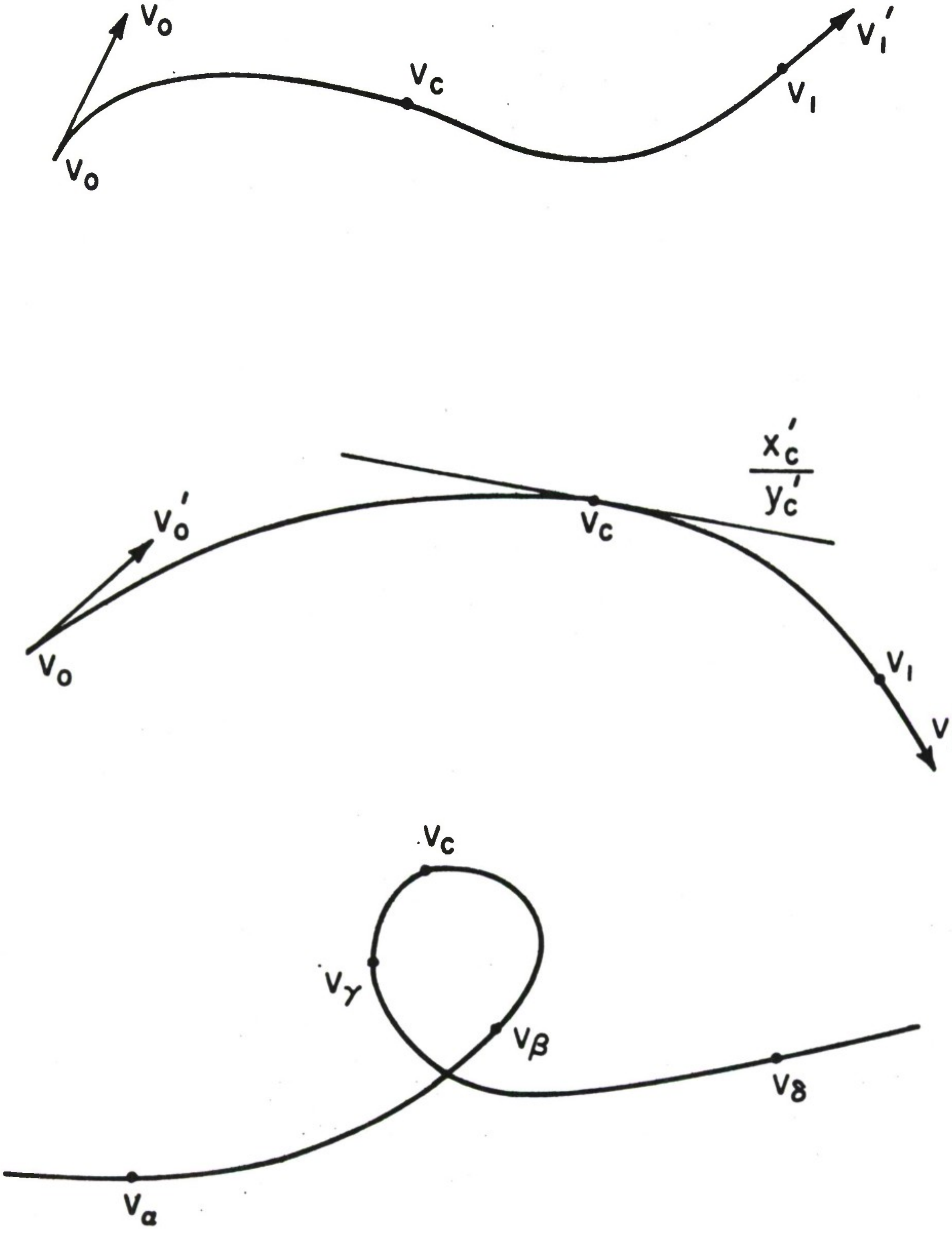


Fig. 2.6.1 Specifying the Curve

an additional condition.* One convenient such condition is to specify that the slope at $\vec{v}_c$ be given by the ratio $\frac{x'_c}{y'_c} = \frac{dx}{dy}$. Taking derivatives with respect to u, we have

$$[h_0 h_1 h'_0 h'_1] \begin{bmatrix} F'_0 & & G'_0 & \\ & F'_1 & & G'_1 \\ G'_0 & & & \\ & G'_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix} = \left.\frac{d(h_c \vec{v}_c)}{du}\right|_{u=u_c} =$$

$$= h'_c \vec{v}_c + h_c \vec{v}'_c = h'_c \vec{v}_c + \vec{v}'_c$$

because we have let $h_c = 1$. ($\vec{v}_c = [x_c y_c 1]$.) Since the third component of $\vec{v}'_c$, $\vec{v}'_0$, $\vec{v}'_1$ is 0 we can solve for

$$h'_c = [h_0 h_1 h'_0 h'_1] \begin{bmatrix} F'_0 \\ F'_1 \\ G'_0 \\ G'_1 \end{bmatrix}$$

and thus

$$[h_0 h_1 h'_0 h'_1] \begin{bmatrix} F'_0 & & G'_0 & \\ & F'_1 & & G'_1 \\ G'_0 & & & \\ & G'_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix} = [h_1 h_1 h'_0 h'_1] \begin{bmatrix} F'_0 \\ F'_1 \\ G'_0 \\ G'_1 \end{bmatrix} \vec{v}_c + \vec{v}'_c$$

or, rearranging,

---

* $\vec{v}_c$ will not be in the range of V if and only if $\vec{v}'_0$, $\vec{v}'_1$ and $\vec{v}_0 - \vec{v}_1$ are collinear and $\vec{v}_c - \vec{v}_1$ is not on the same line. This means that the tangents at each endpoint point to the other end <u>and</u> $\vec{v}_c$ is not on the line between the two endpoints.

$$[h_0 h_1 h'_0 h'_1] \begin{bmatrix} F'_0 & & G'_0 & \\ & F'_1 & & G'_1 \\ G'_0 & & & \\ & G'_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 - \vec{v}_c \\ \vec{v}_1 - \vec{v}_c \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix} = \vec{v}'_c = [x'_c y'_c 0]$$

Denote the left side of the above by Q. Since we are only concerned with the slope at $\vec{v}_c$, we can extract the first and second components by a post-multiplication and obtain the ratio

$$\frac{Q\begin{bmatrix}1\\0\\0\end{bmatrix}}{Q\begin{bmatrix}0\\1\\0\end{bmatrix}} = \frac{x'_c}{y'_c} \qquad \text{whence} \qquad Q\begin{bmatrix}y'_c\\-x'_c\\0\end{bmatrix} = 0$$

If we adjoin this single equation to the original equation for the curve passing through the point $\vec{v}_c$, we have

$$[h_0 h_1 h'_0 h'_1] \left[\begin{bmatrix} F_0 & & G_0 & \\ & F_1 & & G_1 \\ G_0 & & & \\ & G_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 \\ \vec{v}_1 \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix} \middle| \begin{bmatrix} F'_0 & & G'_0 & \\ & F'_1 & & G'_1 \\ G'_0 & & & \\ & G'_1 & & \end{bmatrix} \begin{bmatrix} \vec{v}_0 - \vec{v}_c \\ \vec{v}_1 - \vec{v}_c \\ \vec{v}'_0 \\ \vec{v}'_1 \end{bmatrix} \begin{bmatrix} y'_c \\ -x'_c \\ 0 \end{bmatrix} \right] = [\vec{v}_c | 0]$$

If the matrix is invertible, we can solve for h_0, h_1, h'_0, h'_1 and determine a curve passing a specified point $\vec{v}_c$ with a given slope $\frac{x'_c}{y'_c}$ with given endpoints $\vec{v}_0$, $\vec{v}_1$ and tangent vectors $\vec{v}'_0$, $\vec{v}'_1$ at those endpoints. As with the three-dimensional case above, we could characterize this problem in somewhat more generality in terms of the range of the appropriate matrices, but we will not do so since the conditions are not simple enough to be interesting.

We could in the previous examples have just as easily required the curve to pass through four non-planar points $\vec{v}_\alpha$, $\vec{v}_\beta$, $\vec{v}_\gamma$, $\vec{v}_\delta$ at different values α, β, γ, δ of the parameter. We would have

$$\begin{bmatrix} \vec{\phi}(\alpha) \\ \vec{\phi}(\beta) \\ \vec{\phi}(\gamma) \\ \vec{\phi}(\delta) \end{bmatrix} A = \begin{bmatrix} h_\alpha \vec{v}_\alpha \\ h_\beta \vec{v}_\beta \\ h_\gamma \vec{v}_\gamma \\ h_\delta \vec{v}_\delta \end{bmatrix}$$

or

$$A = \begin{bmatrix} \vec{\phi}(\alpha) \\ \vec{\phi}(\beta) \\ \vec{\phi}(\gamma) \\ \vec{\phi}(\delta) \end{bmatrix}^{-1} \begin{bmatrix} h_\alpha \vec{v}_\alpha \\ h_\beta \vec{v}_\beta \\ h_\gamma \vec{v}_\gamma \\ h_\delta \vec{v}_\delta \end{bmatrix}$$

Then let us ask that the curve pass through $\vec{v}_c$ at $u = u_c$:

$$\vec{\phi}(u_c) M \begin{bmatrix} h_\alpha \vec{v}_\alpha \\ h_\beta \vec{v}_\beta \\ h_\gamma \vec{v}_\gamma \\ h_\delta \vec{v}_\delta \end{bmatrix} = \vec{v}_c$$

where now we have let

$$M = \begin{bmatrix} \vec{\phi}(\alpha) \\ \vec{\phi}(\beta) \\ \vec{\phi}(\gamma) \\ \vec{\phi}(\delta) \end{bmatrix}^{-1}$$

If we define $\vec{\phi}(u_c)M = [F_0 F_1 F_2 F_3]$, we have after rearranging

$$[h_\alpha h_\beta h_\gamma h_\delta] \begin{bmatrix} F_0 \vec{v}_\alpha \\ F_1 \vec{v}_\beta \\ F_2 \vec{v}_\gamma \\ F_3 \vec{v}_\delta \end{bmatrix} = \vec{v}_c$$

which can be solved directly for $[h_\alpha h_\beta h_\gamma h_\delta]$ if the matrix is non-singular, that is, if the four points are non-planar. In this way we have asked the three-dimensional curve to pass through five specific points at five specific values of the parameter. Similar results could be derived in two dimensions.

SECTION III

SURFACES

3.1 INTRODUCTION

Having more or less exhausted the interesting properties of rational parametric cubic curves, we turn to a consideration of a class of similarly defined three-dimensional surfaces. After presenting the basic definition of this class of rational bi-cubic surfaces, this section presents several different but related characterizations of them. In general, we define a surface by means of a $4 \times 4 \times 4 = 64$ element tensor. The difficult problem is to determine the numbers to place in the tensor in order to display a particular desired surface. We will talk about characterizing the surface in terms of its boundary curves, tangent vectors, normal vectors, and points through which the surface is to pass. We do not speak about a complete surface, but rather about patches which may be combined to form a more complete surface; questions of continuity arising in this connection are also discussed.

3.2 SURFACE EQUATION

Let B_{ijk} be a $4 \times 4 \times 4$ array, then the components of the homogeneous coordinates of a point on a surface are given by a vector function $\vec{P}(u, v)$ of two parameters, the two degrees of freedom on the surface, as

$$P_k = u^i B_{ijk} v^j \; ; \qquad 0 \leq u \leq 1, \quad 0 \leq v \leq 1.$$

This expression may be written in matrix notation as

$$P_{hx} = [u^3 u^2 u \, 1] B_{hx} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$P_{hy} = [u^3 u^2 u\, 1] B_{hy} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$P_{hz} = [\text{et cetera}]$$

$$P_h = [\text{et cetera}]$$

These expressions may also be written without subscripts in the following fashion to indicate all four equations, the implication being that B is a matrix whose components are vectors:

$$\vec{P}(u, v) = [u^3 u^2 u\, 1] B \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

We could also define a time-varying surface as

$$P_k(u, v, t) = u^i B_{ijk}(t) v^j$$

where $B_{ijk}(t) = t^\ell T_{ijk\ell}$; this surface would change smoothly from $B_{ijk}(0) = T_{ijk0}$ to $B_{ijk}(1) = \sum_\ell T_{ijk\ell}$ as t changes from 0 to 1. Even more simply, we could define

$$P_k(u, v, t) = u^i [tB_{ijk} + (1-t)B'_{ijk}] v^j$$

in which the surface would change "linearly" from the surface B to the surface B′. Further generalizations are obvious.

Having defined the class of surfaces to be considered, the remainder of this section is concerned with determining the 64 numbers in the tensor that defines the surface, subject to various constraints.

3.3 ENDPOINT DERIVATIVE FORM

Borrowing Coons' notation [7], let F_0, F_1, G_0, G_1 be functions of one variable such that

$$\left.\begin{array}{ll} F_i(j) = \delta^i_j \;; & G_i(j) = 0 \\ F'_i(j) = 0 \;\; ; & G'_i(j) = \delta^i_j \end{array}\right\} \qquad i, j = 0, 1$$

Then we can define a surface as

$$\vec{P}(u, v) = [F_0(u), F_1(u), G_0(u), G_1(u)]\, A \begin{bmatrix} F_0(v) \\ F_1(v) \\ G_0(v) \\ G_1(v) \end{bmatrix}$$

Because of the definition of the F_i, G_i functions, the components of A relate directly to various partial derivatives of the surface function $\vec{P}(u, v)$ as indicated below:

$$A = \begin{bmatrix} 00 & 01 & 00_v & 01_v \\ 10 & 11 & 10_v & 11_v \\ 00_u & 01_u & 00_{uv} & 01_{uv} \\ 10_u & 11_u & 10_{uv} & 11_{uv} \end{bmatrix}$$

where the notation, from Coons, is exemplified by

$$11_{uv} = \left.\frac{\partial\vec{P}(u, v)}{\partial u \partial v}\right|_{\substack{u=1\\v=1}} = \vec{A}^{11}_{uv} \qquad\qquad 01_u = \left.\frac{\partial\vec{P}(u, v)}{\partial u}\right|_{\substack{u=0\\v=1}} = \vec{A}^{01}_u$$

Note well that these derivatives refer to each component of the four-dimensional homogeneous coordinate vector; for example,

$$\vec{P}_u = \frac{\partial(h(u, v)\vec{p}(u, v))}{\partial u} = \frac{\partial h}{\partial u}\,\vec{p} + h\,\frac{\partial\vec{p}}{\partial u}$$

and we must solve for

$$\frac{\partial\vec{p}}{\partial u} = \frac{\vec{P}_u - \vec{p}h_u}{h} \quad \text{where } \vec{p} = [x\,y\,z\,1], \quad \frac{\partial\vec{p}}{\partial u} = [x'\,y'\,z'\,0]$$

to obtain the partial derivatives of the three-dimensional components, x(u, v), y(u, v) and z(u, v).

If we insist on F_i, G_i being cubic polynomials, the conditions define them completely as

$$[F_0(u)F_1(u)G_0(u)G_1(u)] = [u^3u^2\,u\,1]\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The 4 X 4 matrix above occurs often and has become identified with the symbol M, for "magic" matrix. The tensor A in the definition of the surface is thus the representation of a surface using the basis functions F_0, F_1, G_0, and G_1.

We can convert between the two forms of representation thus far introduced -- the polynomial basis and the endpoint derivative basis -- by means of the identities

$$B = MAM^T$$
$$A = M^{-1}BM^{-1^T}$$

since $[F_0F_1G_0G_1] = [u^3u^2\,u\,1]M$. In these cases the matrix multiplication implied in the equation is that obtained by taking each slice of the tensors separately, as $B_{hx} = MA_{hx}M^T$, or, written in indicial form,

$$B_{ijk} = M_{i\ell}\,A_{\ell mk}\,M_{mj}$$

$$A_{ijk} = (M^{-1})_{i\ell}\,B_{\ell mk}(M^{-1})_{mj}$$

3.4 CURVES ASSOCIATED WITH THE SURFACE

By fixing one parameter a curve in the surface is traced by letting the other parameter range over the interval [0, 1]. For example, let v = b, then

$$\vec{P}(u,b) = [u^3 u^2 u\ 1]\, B \begin{bmatrix} b^3 \\ b^2 \\ b \\ 1 \end{bmatrix} = [u^3 u^2 u\ 1]\, A_{ub}$$

where

$$A_{ub} = B \begin{bmatrix} b^3 \\ b^2 \\ b \\ 1 \end{bmatrix}$$

is the matrix representing the curve $\vec{B}^{v=b}$ as a rational parametric cubic.

The curves $\vec{P}(0,v)$, $\vec{P}(1,v)$, $\vec{P}(u,0)$, $\vec{P}(u,1)$ are the boundary curves of the surface; in the shorthand notation they are denoted by 0v, 1v, u0, u1.

We will denote the matrix describing a curve for $u = \text{constant}$ by

$$A_{av} = ([a^3 a^2 a\ 1]\, B)^T$$

so that the curve is given by an expression of the standard form,

$$\vec{P}(a,v) = [v^3 v^2 v\ 1]\, A_{av}$$

For computer display of the surface we generate the family of curves

$$\vec{P}_{ub},\ b = \frac{i}{m}, \qquad i = 0, 1, \ldots, m$$

or the family

$$\vec{P}_{av},\ a = \frac{j}{n}, \qquad j = 0, 1, \ldots, n$$

or both. The continuous curve $\vec{P}_{ub}$ will be approximated by a series of chords

$$\overline{\vec{P}_{u_i b}\vec{P}_{u_{i+1} b}},\ u_i = \frac{i}{k}, \qquad i = 0, 1, \ldots, k-1$$

and $\vec{P}_{av}$ by the chords

$$\overline{\vec{P}_{av_j}\vec{P}_{av_{j+1}}}, \quad v_j = \frac{j}{\ell}, \quad j = 0, 1, \ldots, \ell-1$$

3.5 ITERATIVE DISPLAY OF SURFACE

The successive chords of a curve and the successive curves of a family on the surface are computed using one of the finite difference schemes of subsection 2.2. If we use forward differences, we thus compute

$$P_k(m\delta, n\gamma) = \binom{m}{i} \varepsilon^i C_{ijk} \kappa^j \binom{n}{j}$$

for an appropriate tensor C dependent on the surface and the particular family of curves to be generated. The multiplication by $\binom{m}{i}\varepsilon^i$ is equivalent to multiplication by the matrix

$$\begin{bmatrix} 1 & \varepsilon & 0 & 0 \\ 0 & 1 & \varepsilon & 0 \\ 0 & 0 & 1 & \varepsilon \\ 0 & 0 & 0 & 1 \end{bmatrix}^m$$

which is achieved by simple shifts and additions, and similarly for $\kappa^j\binom{n}{j}$. The summations over i and j are commutative, so if we are drawing the curves $\vec{P}(u, n\gamma)$ we first compute a 4 × 4 matrix $C^n_{ik} = C_{ijk}\kappa^j\binom{n}{j}$ for each curve and then iterate upon it to compute the successive points $P_k(m\delta, n\gamma) = \binom{m}{i}\varepsilon^i C^n_{ik}$ on the curve. We actually compute C^n_{ik} through the difference equation

$$C^{n+1}_{ijk} = C^n_{ijk} + \kappa C^n_{ij+1k}$$

where $C^0_{ijk} = C_{ijk}$ and $C^n_{ik} = C^n_{i0k}$. Similarly, to compute $P_k(m\delta, n\gamma)$, we compute

$$C^{nm+1}_{ik} = C^{nm}_{ik} + \varepsilon C^{nm}_{i+1k}$$

where $C^{n0}_{ij} = C^n_{ij}$ and $P_k(m\delta, n\gamma) = C^{nm}_{0k}$.

The initial tensor of forward differences C_{ijk} is obtained from the tensor representing the surface B_{ijk} by the relation

$$C_{ijk} = D_{i\ell}(\delta, \varepsilon)\, B_{\ell mk}\, D_{jm}(\gamma, \kappa)$$

where the D matrices are obtained from the second expression on page 26.

To draw the other set of curves, $P(m\delta, v)$, we first compute $C^{m}_{jk} = \binom{m}{i} \varepsilon^{i} C_{ijk}$ and iterate to compute $P_k(m\delta, n\gamma) = C^{m}_{jk}\, \kappa^{j} \binom{n}{j}$.

The algorithms for using backwards differences are similar and readily determined from the above and from subsection 2.2.

3.6 TRANSFORMATIONS OF THE SURFACE

In the previous subsections the coordinates of a point $\vec{P}(u, v)$ on the surface have been defined by expressions of the form $P_k(u, v) = U_i(u) S_{ijk} V_j(v)$ where $\vec{U}$ and $\vec{V}$ are vector-valued functions of the parameters. Let T be a projective transformation to be applied to the surface at each point $\vec{P}$ and let $\vec{P}'$ be the transformed point. Then we have

$$\begin{aligned} P'_{\ell} = P_k T_{k\ell} &= (U_i(u) S_{ijk} V_j(v)) T_{k\ell} = \\ &= U_i(u)(S_{ijk} T_{k\ell}) V_j(v) = \\ &= U_i(u) S'_{ij\ell} V_j(v) \end{aligned}$$

where the surface tensor S_{ijk} is replaced by a transformed tensor $S'_{ij\ell} = S_{ijk} T_{k\ell}$. This derived tensor, $S'_{ij\ell}$, will generate the transformed surface, without the need to transform each point to be displayed. In other words, the sixteen vectors $\vec{S}_{ij} = [S_{ij0} S_{ij1} S_{ij2} S_{ij3}]$ transform in the same fashion as points on the surface and can be regarded as a set of sixteen points which define the surface. Notice, of course, that any transformation -- including the identity transformation αI -- of the surface must be applied identically to all sixteen $\vec{S}_{ij}$. These properties are obviously what allow us the right to call the S_{ijk} array a tensor.

Notice in particular that the tensor S_{ijk} may be the set of differences needed to generate the surface through finite difference iterations. Transformations of the surface can thus be made after the display representation has been chosen and need not be applied to the abstract representation B_{ijk}.

3.7 REPARAMETERIZATION

As with a curve matrix, the tensor describing a surface can be reparameterized to

1. change the rate at which the surface is traversed by the parameters

and to

2. display other (smaller or larger) portions of the surface with the same parameter range, $0 \leqq u \leqq 1$, $0 \leqq v \leqq 1$.

In the case of surfaces, 1 is particularly important for such a reparameterization changes the appearance of the curvilinear net used to display the surface, by changing the density of contour lines non-uniformly, whereas with curves it affects only the accuracy with which the chords approximate the curve.

For each of the basis functions $[u^3 u^2 u\, 1]$, $[F_0 F_1 G_0 G_1]$ or $[\binom{u}{0} \binom{u}{1}\delta \binom{u}{2}\delta^2 \binom{u}{3}\delta^3]$, there exists a reparameterization matrix $S(\alpha, \beta, r)$, a function of three parameters α, β, and r which will

1. map $u = 0$ to $u = \alpha$
2. map $u = 1$ to $u = \beta$
3. change the parameter rate by a factor r.

These matrices are obtained as described in subsection 2.4.

Two of these reparameterization matrices, $S(\alpha, \beta, r)$ and $S(\gamma, \delta, s)$ can be applied to a surface tensor T to compute a new surface tensor T′ as

$$T'_{ijk} = S_{i\ell}(\alpha, \beta, r)\, T_{\ell mk} S_{jm}(\gamma, \delta, s)$$

in which the parameterization in both u and v has been altered.

A shape-invariant transformation of the form S(0, 1, r) combined with the identity αI has the property of allowing us to adjust the fourth homogeneous coordinate h at three of the corner points at will; in particular, we could arbitrarily assign $h^{00} = h^{01} = h^{10} = 1$ without any loss of generality in the class of surfaces represented. Correspondingly, if we are given a surface constrained in this fashion we can reparameterize it so as to arrive at a more satisfactory rate of display.

To make this arbitrary assignment of values to the homogeneous coordinates, recall that by the results of subsection 2.4 we can adjust the homogeneous coordinates at the ends of a curve at will. Suppose we have a tensor A_{ijk} representing the surface A. Applying a reparameterization matrix S(r) to u we obtain $B_{ijk} = S_{i\ell}(r)A_{\ell jk}$ in which at the corners we have

$$\begin{aligned} B_h^{00} &= r^3 A_h^{00} \\ B_h^{01} &= r^3 A_h^{01} \\ B_h^{10} &= A_h^{10} \\ B_h^{11} &= A_h^{11} \end{aligned}$$

Then applying a reparameterization S(s) to v, we obtain

$$C_{ijk} = B_{imk}S_{jm}(s) = S_{i\ell}(r)A_{\ell mk}S_{jm}(s)$$

in which

$$\begin{aligned} C_h^{00} &= s^3 B_h^{00} = s^3 r^3 A_h^{00} \\ C_h^{01} &= B_h^{01} = r^3 A_h^{01} \\ C_h^{10} &= s^3 B_h^{10} = s^3 A_h^{10} \\ C_h^{11} &= B_h^{11} = A_h^{11} \end{aligned}$$

and lastly applying a complete rescaling $D_{ijk} = \alpha C_{ijk}$, we have

$$D_h^{00} = \alpha s^3 r^3 A_h^{00}$$

$$D_h^{01} = \alpha r^3 A_h^{01}$$

$$D_h^{10} = \alpha s^3 A_h^{10}$$

$$D_h^{11} = \alpha A_h^{11}$$

If none of the D_h or A_h are zero, we can obviously pick α, s, r in order to assign arbitrary values to the coordinate h at three of the corners; the value of h at the fourth corner will then be determined. There are thus a triply infinite number of tensor representations of a surface patch.

3.8 CONSTRUCTING THE SURFACE

The preceding subsections describe some of the properties of a given surface and are thus analytical; the usual question, however, is the constructive problem of generating a surface from external conditions. This and the following two subsections sketch some partial solutions that are being investigated.

Suppose we are given the four boundary curves of a surface, specified by matrices A^{0v}, A^{1v}, A^{u0}, A^{u1} in endpoint derivative form. By the geometrical closure of the curve segments bounding the surface, these matrices are guaranteed to represent the same corner points $\vec{P}(0,0)$, $\vec{P}(0,1)$, $\vec{P}(1,0)$ and $\vec{P}(1,1)$ at the intersections of the generated boundary curves. This representation is, however, in homogeneous coordinates and we have no guarantee that the four-dimensional vectors given by the matrices are the same. (In other words, $\vec{A}^{0v}(v=0)$ is proportional to but not necessarily equal to $\vec{A}^{u0}(u=0)$.) By reparameterizing each of the boundary curves separately, we can assign absolute values to the homogeneous coordinate h at each of the corners, adjusting the curve matrices A^{0v}, A^{1v}, A^{u0}, A^{u1} so that the endpoints will now be represented by identical vectors.

That is, we adjust

$$A^{u0} \quad \text{so that} \quad A_h^{u0}(u=0) = h^{00}, \quad A_h^{u0}(u=1) = h^{10}$$

$$A^{u1} \quad \text{so that} \quad A_h^{u1}(u=0) = h^{01}, \quad A_h^{u1}(u=1) = h^{11}$$

$$A^{0v} \quad \text{so that} \quad A_h^{0v}(v=0) = h^{00}, \quad A_h^{0v}(v=1) = h^{01}$$

$$A^{1v} \quad \text{so that} \quad A_h^{1v}(v=0) = h^{10}, \quad A_h^{1v}(v=1) = h^{11}$$

In assigning the four values of h, only one degree of freedom will affect the shape of the surface. The other three will be used in specifying the parameterizations in u and v and in assigning an absolute homogeneous scale to the surface. We could thus arbitrarily assign $h^{00} = h^{01} = h^{10} = 1$ and solve for h^{11} according to some additional criterion.

At this point we have filled in the tensor as shown in Figure 3.8.1. (For example, $S_{000} = P_{hx}^{00} = A_{00}^{u0}$; $S_{302} = (P_v^{01})_{hy} = A_{23}^{0v}$.) The remaining sixteen components can be determined from any sixteen independent conditions. These sixteen as yet unspecified components have come to be called the <u>twist</u> partition of the tensor since they refer to mixed partial derivatives, $\frac{\partial^2}{\partial u \partial v}$. A possible set of conditions would be to specify $\vec{p}_{uv}$ at each of the four corners and to specify a point $\vec{p}_s$ through which the surface must pass at some specified value of the parameters, say $u = \frac{1}{2}$, $v = \frac{1}{2}$. With these conditions, we can solve for $(P_h)_{uv}$ at the four corners with one degree of freedom left, say to specify the h value reached at the center points $\vec{P}_s$ or some other appropriate geometrical constraint.

Instead of specifying $\vec{p}_{uv}$ directly, one could specify

$$\left.\frac{\partial \vec{p}}{\partial u}\right|_{\substack{u=0\\v=a}} \qquad \left.\frac{\partial \vec{p}}{\partial u}\right|_{\substack{u=1\\v=b}} \qquad \left.\frac{\partial \vec{p}}{\partial v}\right|_{\substack{u=c\\v=0}} \qquad \left.\frac{\partial \vec{p}}{\partial v}\right|_{\substack{u=d\\v=1}} \qquad \text{(see Figure 3.8.2)}$$

as desired outward tangents from the boundary curves, solving for the mixed partial derivatives.

In all the above we are talking about conditions expressed in terms

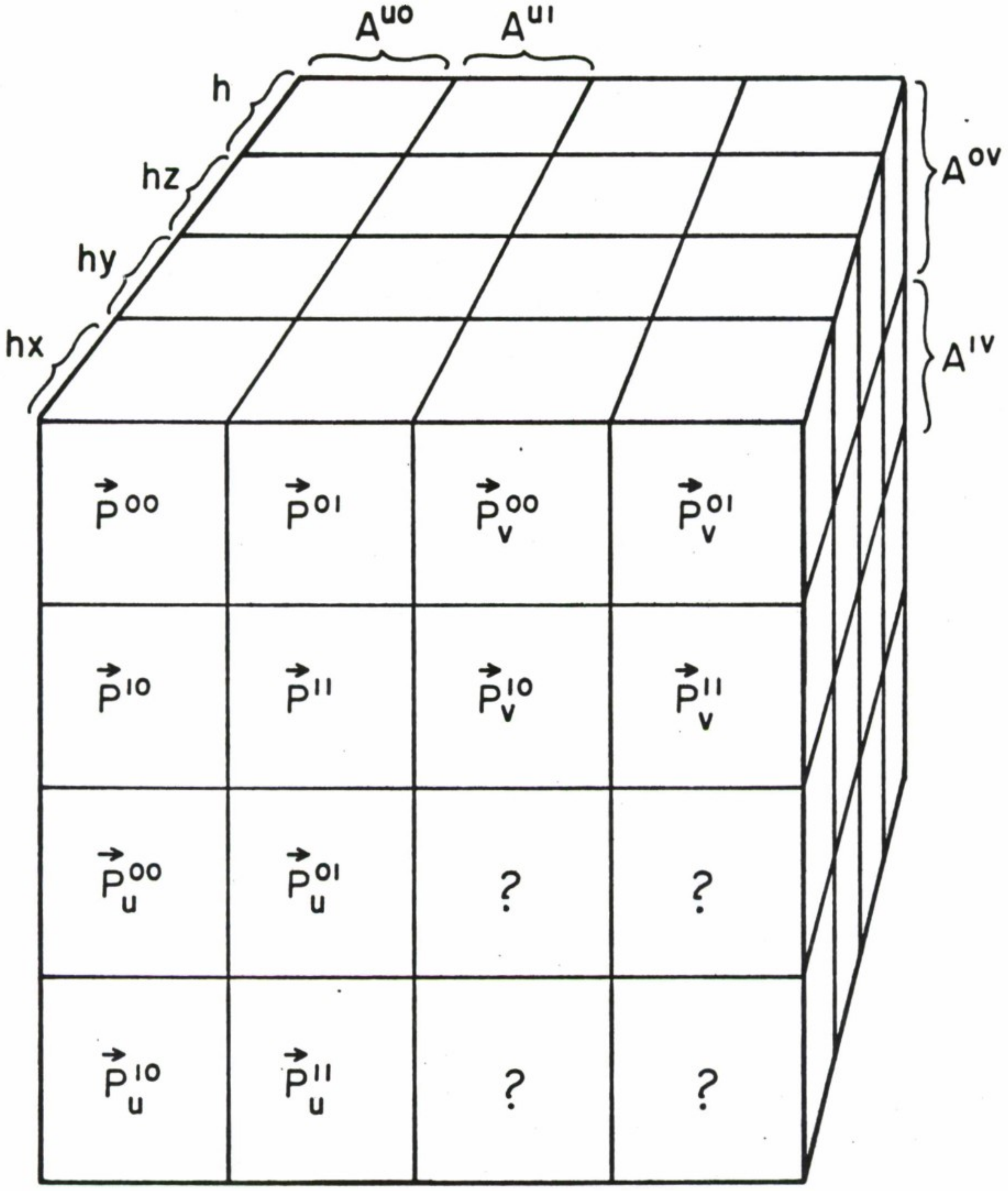
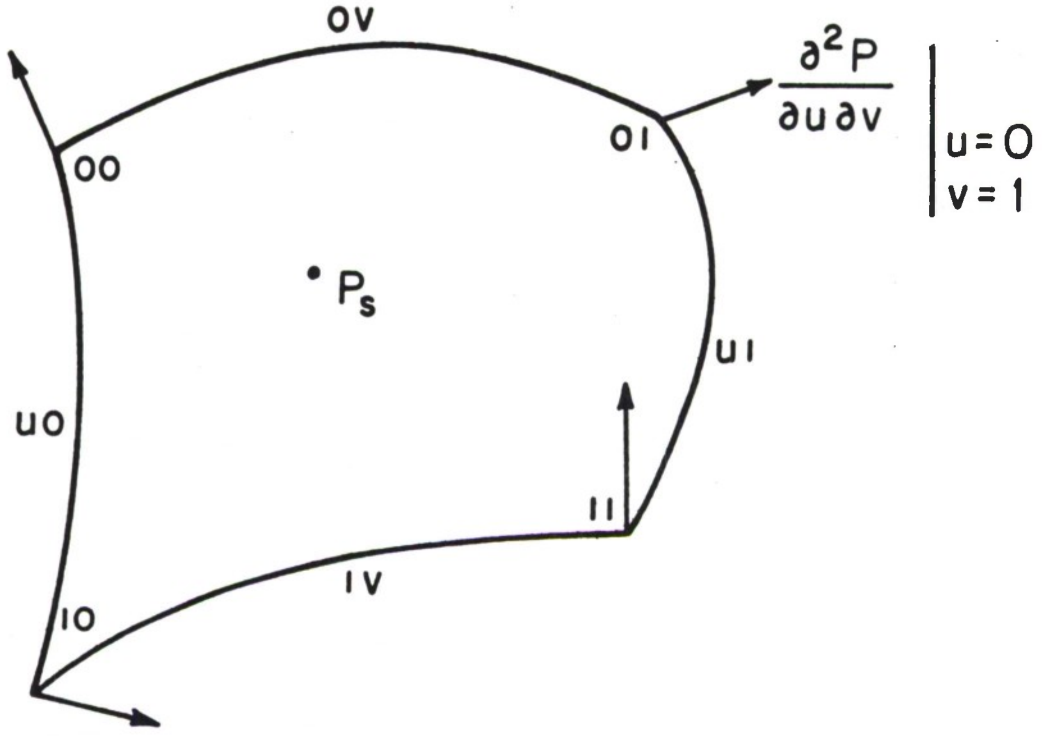


Fig. 3.8.1 Constructing the Surface from Boundary Curves


0V
00
01
$\frac{\partial^2 P}{\partial u \partial v}\Big|_{u=0,\, v=1}$
P_s
U1
U0
11
1V
10

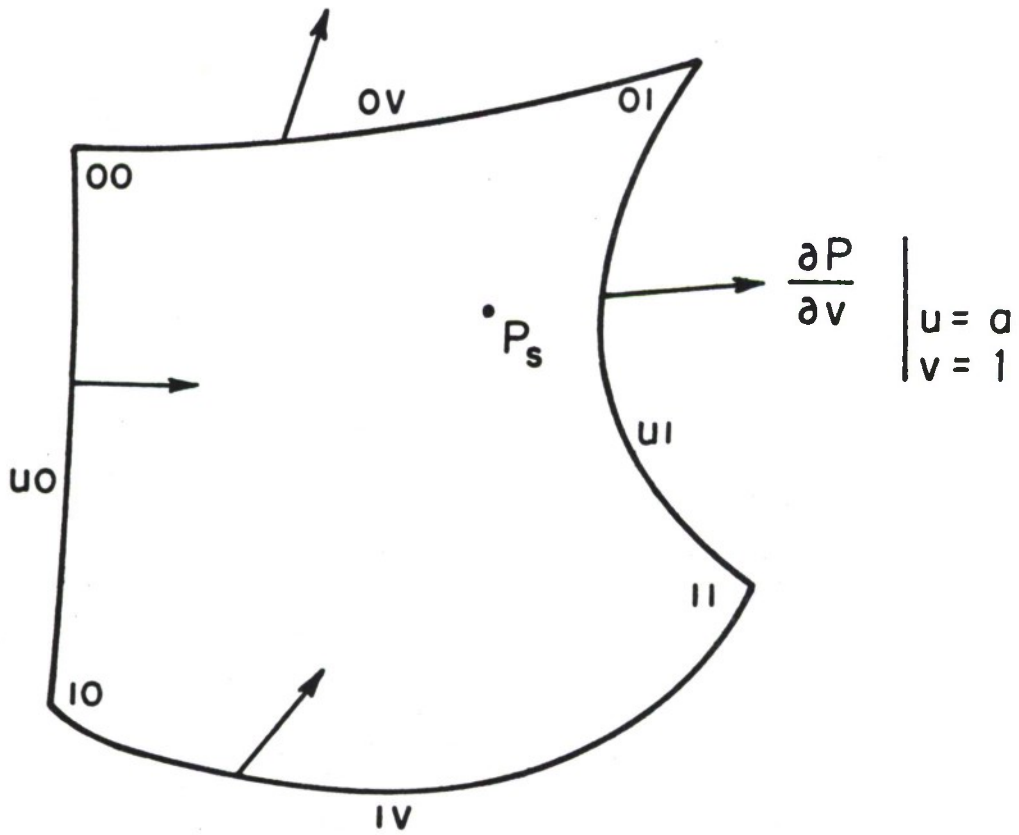


Fig. 3.8.2 Derivative Vectors as Conditions on the Surface

of three-dimensional points or vectors; the determination of the homogeneous coordinate representation for each such vector is done by specifying other conditions, namely, the point $\vec{p}_s$.

Let us examine a particularly elegant way of determining the twist partition of the surface tensor. In this analysis we will use several vectors and matrices whose components are themselves vectors, such as the point 00 or the derivative 01_{uv}.

Let us require that a curve av given by the matrix $[a0\ a1\ a0_v\ a1_v]$ in endpoint derivative form to lie across the surface at the constant parameter u=a as in Figure 3.8.3. Notice that as long as we pick the points a0 and a1 to lie respectively on u0 and u1 at the same value a of the parameter u, the curve av can be a completely arbitrary rational cubic, since by reparameterization we can adjust the homogeneous scale factors at a0 and a1 to correspond precisely to the values assumed by the curves u0 and u1 at the corresponding points.

We thus have

$$[a0\ a1\ a0_v\ a1_v] = [F_0(a)\ F_1(a)\ G_0(a)\ G_1(a)] \begin{bmatrix} 00 & 01 & 00_v & 01_v \\ 10 & 11 & 10_v & 11_v \\ 00_u & 01_u & 00_{uv} & 01_{uv} \\ 10_u & 11_u & 10_{uv} & 11_{uv} \end{bmatrix}$$

We know that the left two terms [a0 a1] of the above equation will automatically hold since we have selected the points a0, a1 to lie on the boundaries and have adjusted the parameterizations properly. The remaining two terms give upon rearranging,

$$[a0_v\ a1_v] = [F_0(a)\,F_1(a)] \begin{bmatrix} 00_v & 01_v \\ 10_v & 11_v \end{bmatrix} + [G_0(a)\,G_1(a)] \begin{bmatrix} 00_{uv} & 01_{uv} \\ 10_{uv} & 11_{uv} \end{bmatrix}$$

which further gives

$$[G_0(a)\ G_1(a)] \begin{bmatrix} 00_{uv} & 01_{uv} \\ 10_{uv} & 11_{uv} \end{bmatrix} = [a0_v\ a1_v] - [F_0(a)\ F_1(a)] \begin{bmatrix} 00_v & 01_v \\ 10_v & 11_v \end{bmatrix}$$

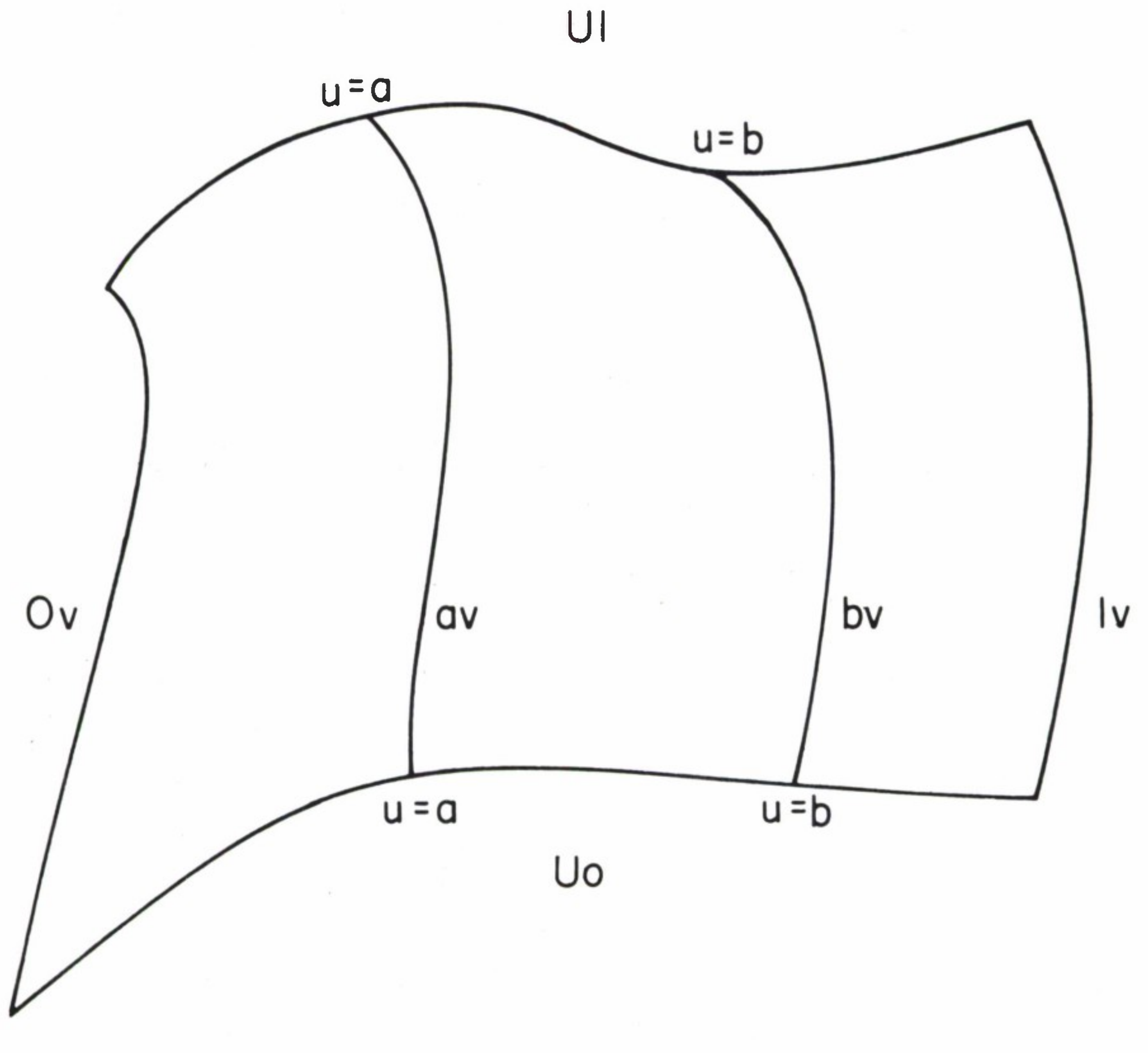


FIG. 3.8.3 INTERIOR CURVES AS CONDITIONS ON THE SURFACE

Let us now presume another curve bv = $[b0\ b1\ b0_v\ b1_v]$ to be chosen on the surface, $b \neq a$. By a precisely similar argument, we obtain the equation

$$[G_0(b)\ G_1(b)]\begin{bmatrix} 00_{uv} & 01_{uv} \\ 10_{uv} & 11_{uv} \end{bmatrix} = [b0_v\ b1_v] - [F_0(b)\ F_1(b)]\begin{bmatrix} 00_v & 01_v \\ 10_v & 11_v \end{bmatrix}$$

Combining the two equations finally gives

$$\begin{bmatrix} G_0(a) & G_1(a) \\ G_0(b) & G_1(b) \end{bmatrix}\begin{bmatrix} 00_{uv} & 01_{uv} \\ 10_{uv} & 11_{uv} \end{bmatrix} = \begin{bmatrix} a0_v & a1_v \\ b0_v & b1_v \end{bmatrix} - \begin{bmatrix} F_0(a) & F_1(a) \\ F_0(b) & F_1(b) \end{bmatrix}\begin{bmatrix} 00_v & 01_v \\ 10_v & 11_v \end{bmatrix}$$

Since a or b are not 0 or 1 and are distinct, the 2×2 matrix

$\begin{bmatrix} G_0(a) & G_1(a) \\ G_0(b) & G_1(b) \end{bmatrix}$ is non-singular and we can solve directly for the twist

partition as

$$\begin{bmatrix} 00_{uv} & 01_{uv} \\ 10_{uv} & 11_{uv} \end{bmatrix} = \begin{bmatrix} G_0(a) & G_1(a) \\ G_0(b) & G_1(b) \end{bmatrix}^{-1}\left[\begin{bmatrix} a0_v & a1_v \\ b0_v & b1_v \end{bmatrix} - \begin{bmatrix} F_0(a) & F_1(a) \\ F_0(b) & F_1(b) \end{bmatrix}\begin{bmatrix} 00_v & 01_v \\ 10_v & 11_v \end{bmatrix}\right]$$

This result allows us to completely characterize a rational bicubic surface in terms of six essentially arbitrary rational cubic curves -- the four boundaries and a pair of curves passing from one boundary to the opposite boundary. (Naturally, the curves are not completely arbitrary -- the boundary curves must close and the two interior curves must terminate at appropriate points on the boundaries.) There is remaining one degree of freedom -- the h^{11} component mentioned earlier -- which could be used to make the surface pass through an arbitrary point, but at an unspecifiable value of the parameters u and v, perhaps even outside of the range $0 < u < 1$, $0 < v < 1$. Solving for the value of h^{11} to satisfy this condition would not be easy, since even finding whether a given surface passes through a given point is a difficult question; numerical approaches would probably be best.

3.9 PRODUCT SURFACES AND SURFACES OF REVOLUTION

A convenient way to specify certain surfaces is as the "product" of two curves; if the curves are chosen appropriately, a surface of revolution results. Let $\vec{P}(u)$ and $\vec{Q}(v)$ be two curves; then the i'th component of a surface S can be defined as

$$S_i = P_i(u)Q_i(v)$$

If we let

$$P_k = \phi_i(u)A_{ik}$$
$$Q_k = \phi_j(v)B_{jk}$$

where A, B are the appropriate matrices for the basis functions $\vec{\phi}$, then $S_k = \phi_i(u)A_{ik}B_{jk}\phi_j(v)$ and we can identify

$$T_{ijk} = A_{ik}B_{jk}$$

as the tensor representing the surface.

If one of the curves $\vec{P}(u)$ or $\vec{Q}(v)$ is planar, say $\vec{P}(u)$, we prove that all curves of the corresponding family, $\vec{S}(u, a)$, are planar:

Let $\vec{A}$ be the vector representing the plane of $\vec{P}(u)$; define a vector $\vec{B}$, component by component, as

$$B_i(v) = \frac{A_i}{Q_i(v)}$$

Then

$$S_i(u, v)B_i(v) = P_i(u)Q_i(v)B_i(v) = P_i(u)A_i = 0$$

where now the summation convention holds, and the curve $\vec{S}(u, a)$ lies in the plane $\vec{B}(a)$. Hence, if both curves $\vec{P}(u)$, $\vec{Q}(v)$ are planar, then all the constant parameter contours are plane sections of the surface.

In particular, let $\vec{P}(u)$ be a curve in the $x = y$ plane and let $\vec{Q}(v)$ be a portion of a circle with center $[0, 0, 1]$ in the $z = 1$ plane. Then we have, in homogeneous coordinates,

$$P_h P_x = P_h P_y$$

$$(Q_h Q_x)^2 + (Q_h Q_y)^2 = (Q_h r)^2 \; ; \quad Q_h Q_z = Q_h$$

as conditions on the curves. The surface, in homogeneous coordinates, is

$$h = P_h Q_h$$
$$hx = P_h P_x Q_h Q_x$$
$$hy = P_h P_y Q_h Q_y = P_h P_x Q_h Q_y$$
$$hz = P_h P_z Q_h Q_z = P_h P_z Q_h$$

or, in regular coordinates,

$$x = P_x Q_x$$
$$y = P_x Q_y$$
$$z = P_z$$

and we have

$$\left(\frac{x}{P_x}\right)^2 + \left(\frac{y}{P_y}\right)^2 = r^2$$

or

$$x^2 + y^2 = (P_x r)^2$$

and also, $z = P_z$, giving a family of circles about the z axis. All plane sections perpendicular to the z axis will be circles and all plane sections passing through the z axis will have the same shape as the original curve $\vec{P}(u)$. Hence, S is part of a surface of revolution constructed by rotating the curve $\vec{P}(u)$ about the z axis.

3.10 CONTINUITY CONDITIONS

In constructing an object from an assemblage of elementary surface patches, it is often necessary to enforce certain degrees of continuity at the junction between two patches. In particular, suppose

S and T are two surfaces meeting in the common boundary curve B(u), with the other boundaries as indicated in Figure 3.10.1. Assume the parameterization in u and the homogeneous scale have been adjusted so that $\vec{S}(u,1) = \vec{T}(u,0) = \vec{B}(u)$, which is always possible by the results of subsection 2.4. Suppose we wish the following constraints to be satisfied:

1. The 0v, 1v boundaries have a continuous tangent direction at P and Q. That is, the 0v, 1v boundary curves are continuous and have a continuous tangent line.

2. Everywhere along B, the two surfaces have a common tangent plane. That is, the surfaces are continuous and have a continuous unit normal vector.

It will turn out that the second constraint implies the first, so we will not explicitly enforce the first constraint.

In order to talk about the tangent planes to a surface, observe the following result.

$$\vec{T}_u = \frac{\partial(h\vec{t})}{\partial u} = h_u\vec{t} + h\vec{t}_u = h_u\left(\vec{t} + \frac{h}{h_u}\vec{t}_u\right)$$

But $\vec{t} + \frac{h}{h_u}\vec{t}_u$ is a point on the tangent $\frac{\partial\vec{t}}{\partial u}$ so $\vec{T}_u$ is a point on that tangent. Similarly, $\vec{T}_v$ is a point on the tangent $\frac{\partial\vec{t}}{\partial v}$. Therefore, a point $\vec{P}$ on the tangent plane at $\vec{T}$ must be in the plane through $\vec{T}_u$, $\vec{T}_v$ and $\vec{T}$, that is, $|\vec{P}\,\vec{T}\,\vec{T}_u\,\vec{T}_v| = 0$.

From this result, the relation that tangent planes to surfaces S and T coincide can be expressed as

$$|\vec{S}\,\vec{T}\,\vec{T}_u\,\vec{T}_v| = |\vec{S}_u\,\vec{T}\,\vec{T}_u\,\vec{T}_v| = |\vec{S}_v\,\vec{T}\,\vec{T}_u\,\vec{T}_v| = 0$$

where the vectors are evaluated at the same value of the parameter u on the curve B. We derive this by noting that all the points $\vec{S}$, $\vec{S}_u$, $\vec{S}_v$ must be on the tangent plane to S and hence must also be on the tangent plane to T. By our initial assumption $\vec{S}^{u1} = \vec{T}^{u0}$ and hence $\vec{S}_u^{u1} = \vec{T}_u^{u0}$. Thus the first two determinants vanish so we consider the third.

In general, we cannot expect $\vec{S}_v$ to bear any relation to $\vec{T}$ or $\vec{T}_u$

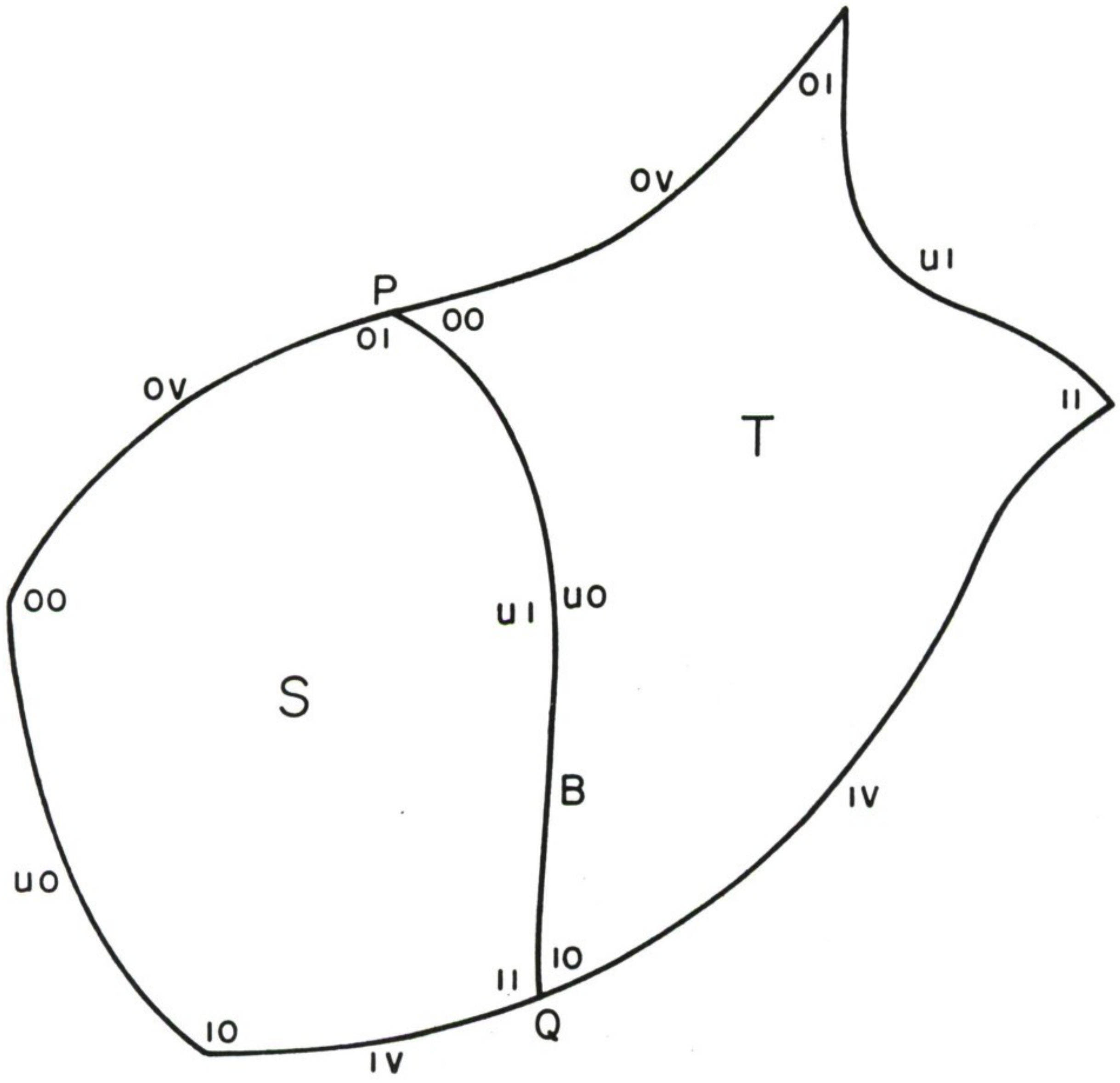


Fig. 3.10.1 Adjacent Surface Continuity Conditions

so the vanishing of the third determinant requires $\vec{S}_v$ to be proportional to $\vec{T}_v$, for all u. But this relation is precisely the same as asking that the function $\vec{S}_v(u)$ in representing a curve represent the same curve as $\vec{T}_v(u)$, regarding the derivatives $\frac{\partial}{\partial v}$ as the homogeneous coordinates of the points on some curve in three dimensions. Since we have assumed equal parameterization along the boundary curve B, this requires

$$\vec{S}_v^{01} = k\vec{T}_v^{00}$$

$$\vec{S}_v^{11} = k\vec{T}_v^{10}$$

$$\vec{S}_{uv}^{01} = k\vec{T}_{uv}^{00}$$

$$\vec{S}_{uv}^{11} = k\vec{T}_{uv}^{10}$$

for an arbitrary proportionality constant k. We deduce this by noting from the results of subsection 2.4 that a curve P(u) is the same as the curve Q(u) with the point P(u) corresponding to the point Q(u) if and only if $\vec{P}(u) = k\vec{Q}(u)$, for all u, for some constant k.

If we now apply an arbitrary reparameterization in u, we obtain a more general set of sufficient conditions that two surfaces S and T share a common boundary and satisfy conditions 1 and 2 as

$$\begin{bmatrix} \vec{S}^{01} & \vec{S}_v^{01} \\ \vec{S}^{11} & \vec{S}_v^{11} \\ \vec{S}_u^{01} & \vec{S}_{uv}^{01} \\ \vec{S}_u^{11} & \vec{S}_{uv}^{11} \end{bmatrix} = \alpha \begin{bmatrix} r^3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3(1-r)r^2 & 0 & r^2 & 0 \\ 0 & 3(1-r) & 0 & r \end{bmatrix} \begin{bmatrix} \vec{T}^{00} & k\vec{T}_v^{00} \\ \vec{T}^{10} & k\vec{T}_v^{10} \\ \vec{T}_u^{00} & k\vec{T}_{uv}^{00} \\ \vec{T}_u^{10} & k\vec{T}_{uv}^{11} \end{bmatrix}$$

where $\alpha = r = 1$ will give equal parameterization along the boundary curve.

Please notice that, as usual, in all the above we are talking about homogeneous coordinate vectors and that an equation of the form

$\vec{S}_v^{01} = k\vec{T}_v^{00}$ does not constrain the tangent vectors in ordinary three dimensions to be equal. In fact, we have

$$(S_h \vec{s}) = (S_h)_v \vec{s} + S_h \vec{s}_v = k(T_h \vec{t})_v = k(T_h)_v \vec{t} + k T_h \vec{t}_v$$

where $\vec{s} = [x\,y\,z\,1]$ and similarly for $\vec{t}$. Now $\vec{s} = \vec{t}$ but in general $\vec{S}_h \neq \vec{T}_h$ so we have

$$(S_h)_v \vec{s} + S_h \vec{s} = k(T_h)_v \vec{s} + k T_h \vec{t}$$

or

$$\{(S_h)_v - k(T_h)_v\} \vec{s} = k T_h \vec{t}_v - S_h \vec{s}_v$$

The fourth coordinate of this vector equation gives

$$(S_h)_v - k(T_h)_v = 0$$

and thus

$$0 = k T_h \vec{t}_v - S_h \vec{s}_v$$

or

$$\vec{t}_v = \frac{S_h}{kT_h} \vec{s}_v$$

Although there is tangent vector direction continuity across the boundary, there is an arbitrary magnitude discontinuity, controlled by the constant k.

Notice that this specific relation (along with the corresponding one for $\vec{S}_v^{11}$) forces boundary tangent vector direction continuity at P and Q, satisfying condition 1.

SECTION IV

IMPLEMENTATION

4.1 EXAMPLES

Figures 4.1.1 through 4.1.5 have been included to illustrate the appearance of the surfaces. The figures are all true perspective projections of the surfaces from varying vantage points. They should be viewed from such a distance as to give a 45° field of view. They were computed upon a DEC PDP-1 with DEC 340 scopes used for display. The PDP-1 is a rather ancient machine by modern standards, having an 18-bit word length, 5-μsec cycle time, 25-μsec multiply, 40-μsec divide and no floating point.

The time to compute a new view of a surface is 1 to 2 seconds for these figures, depending on the number of curves used. This is about 1-2 ms. per chord. Each curve is approximated by 32 chords.

The tensors for the cylinder, sphere and toroid were manually computed as surfaces of revolution. The hyperbolic paraboloid or saddle surface is simply the equation $z = x^2 - y^2$ expressed parametrically in the tensor. The surface shown in Figure 4.1.5 was obtained by making a 20% random variation in each of the components of the tensor for the cylinder of Figure 4.1.2.

4.2 EXPERIMENTAL IMPLEMENTATION

Faced with the speed and capacity restrictions of the available interactive computing facilities (the PDP-1 graphics research laboratory), we decided that the best way to illustrate the capabilities of this material would be the experimental production of a short motion-picture film showing representative examples of the surfaces moving about to give a good impression of their three-dimensional structure and appearance. This section comments upon the results of that experiment, with the material in Appendix II proposing in relation to this section what we would like to have tried, given a sufficiently more powerful interactive environment.

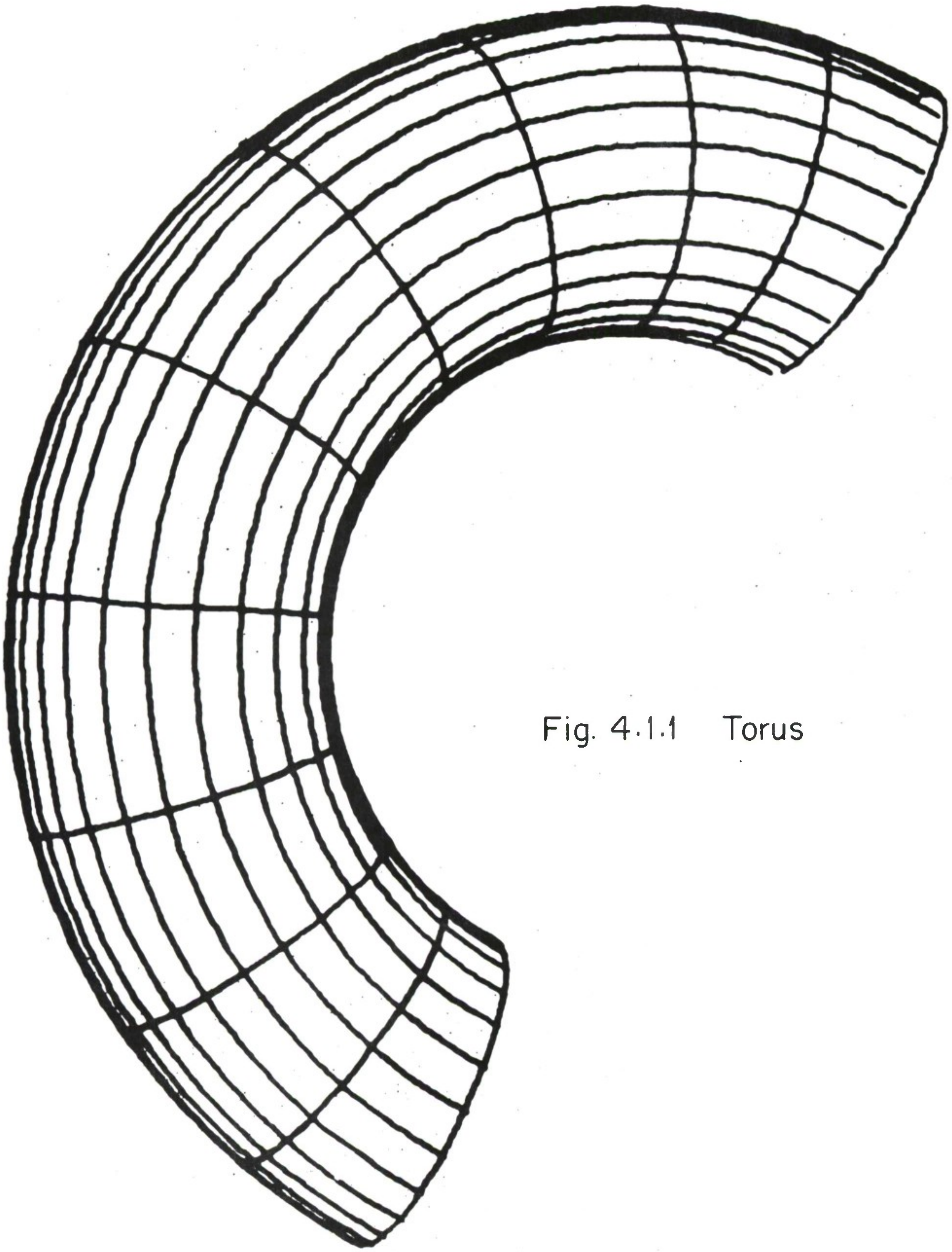


Fig. 4.1.1 Torus

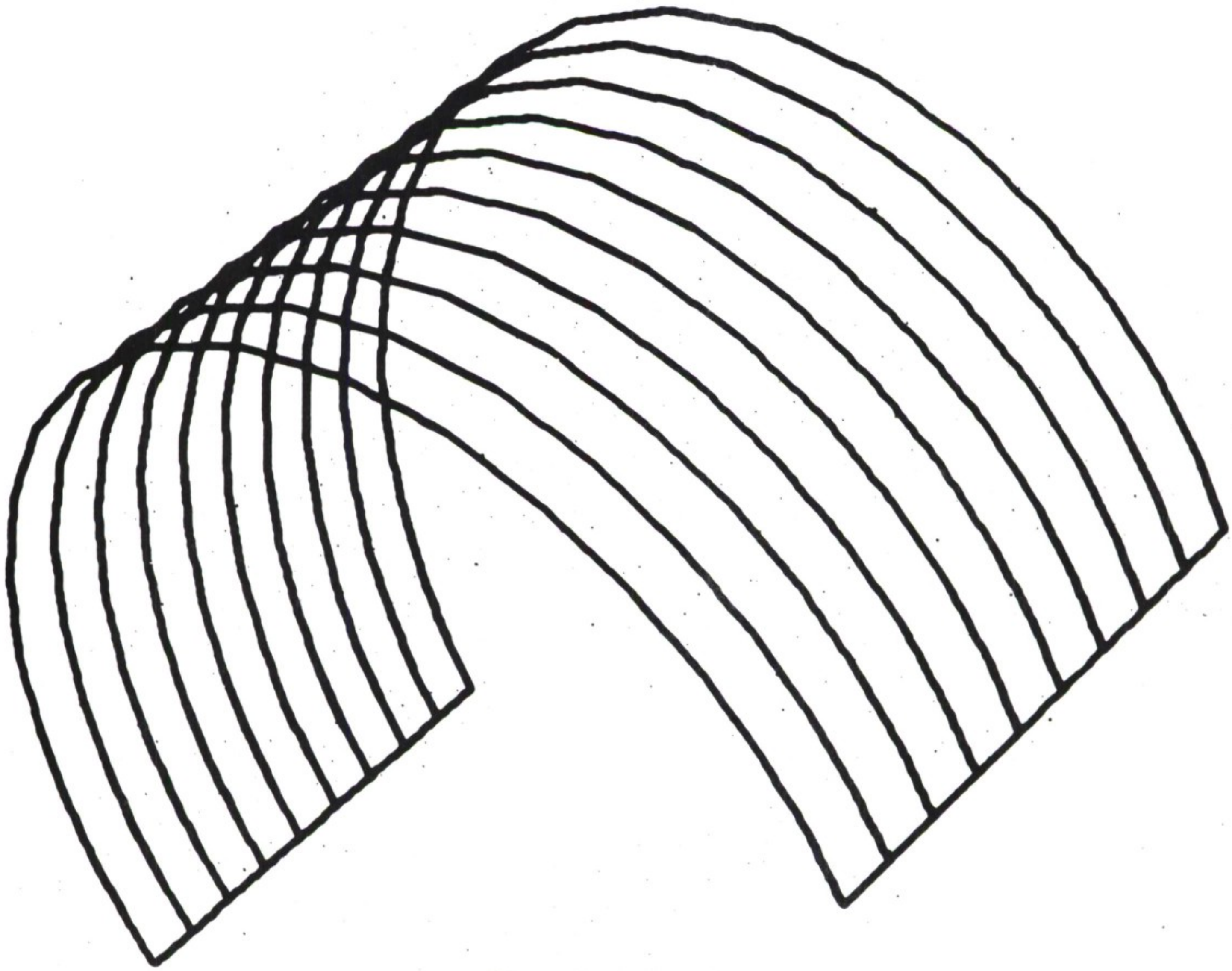


Fig. 4.1.2 Cylinder

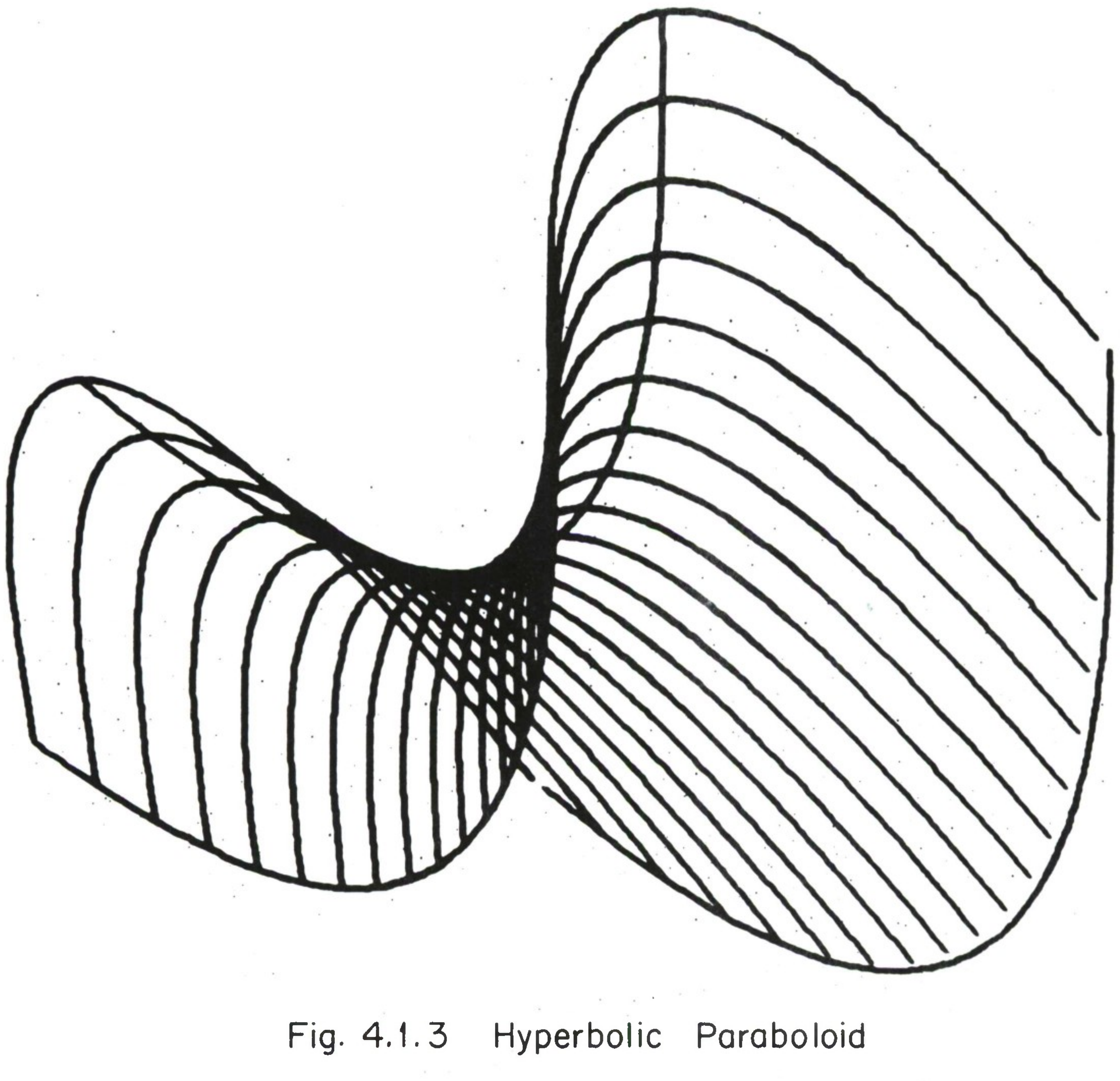


Fig. 4.1.3 Hyperbolic Paraboloid

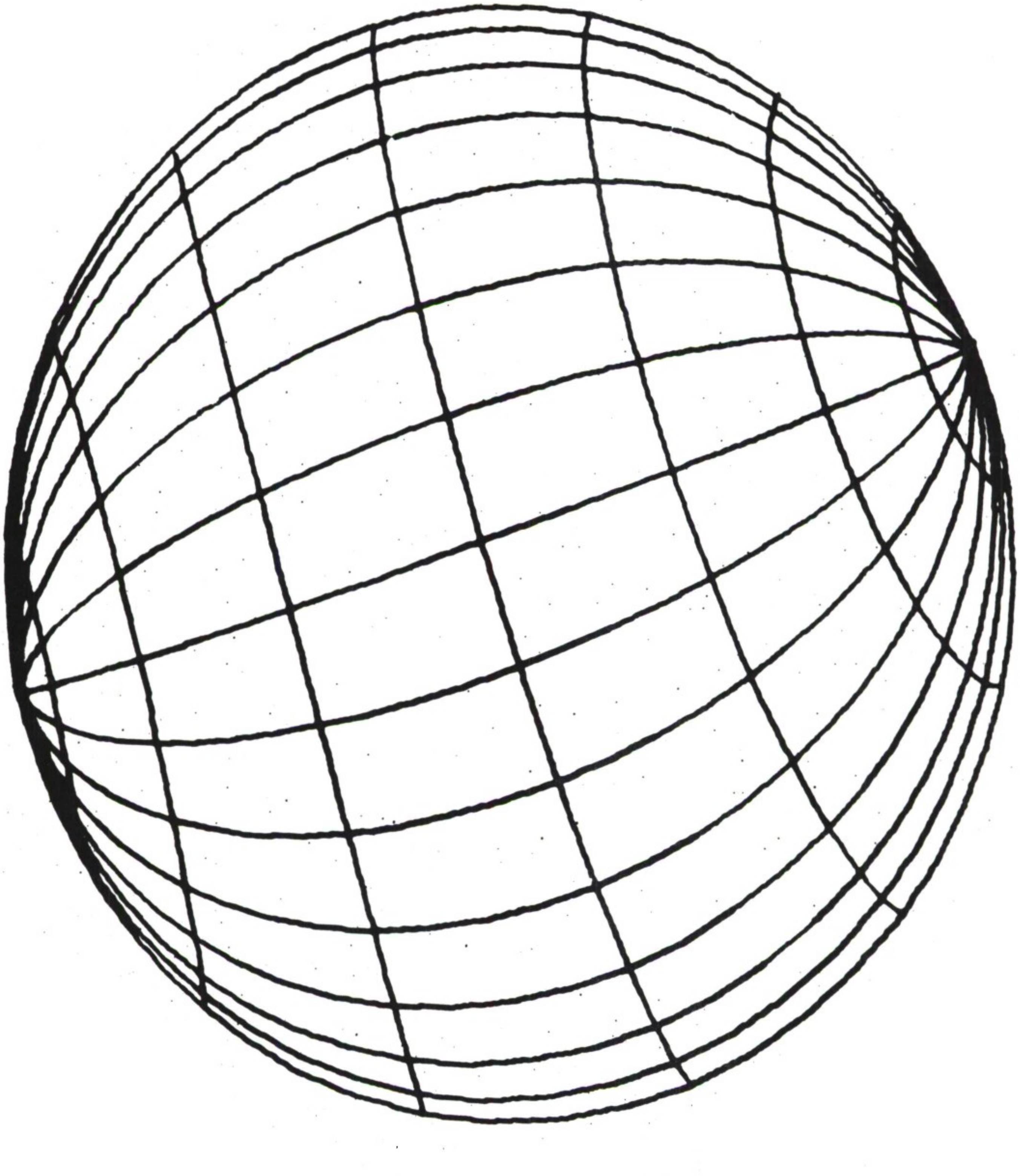


Fig. 4.1.4 Sphere

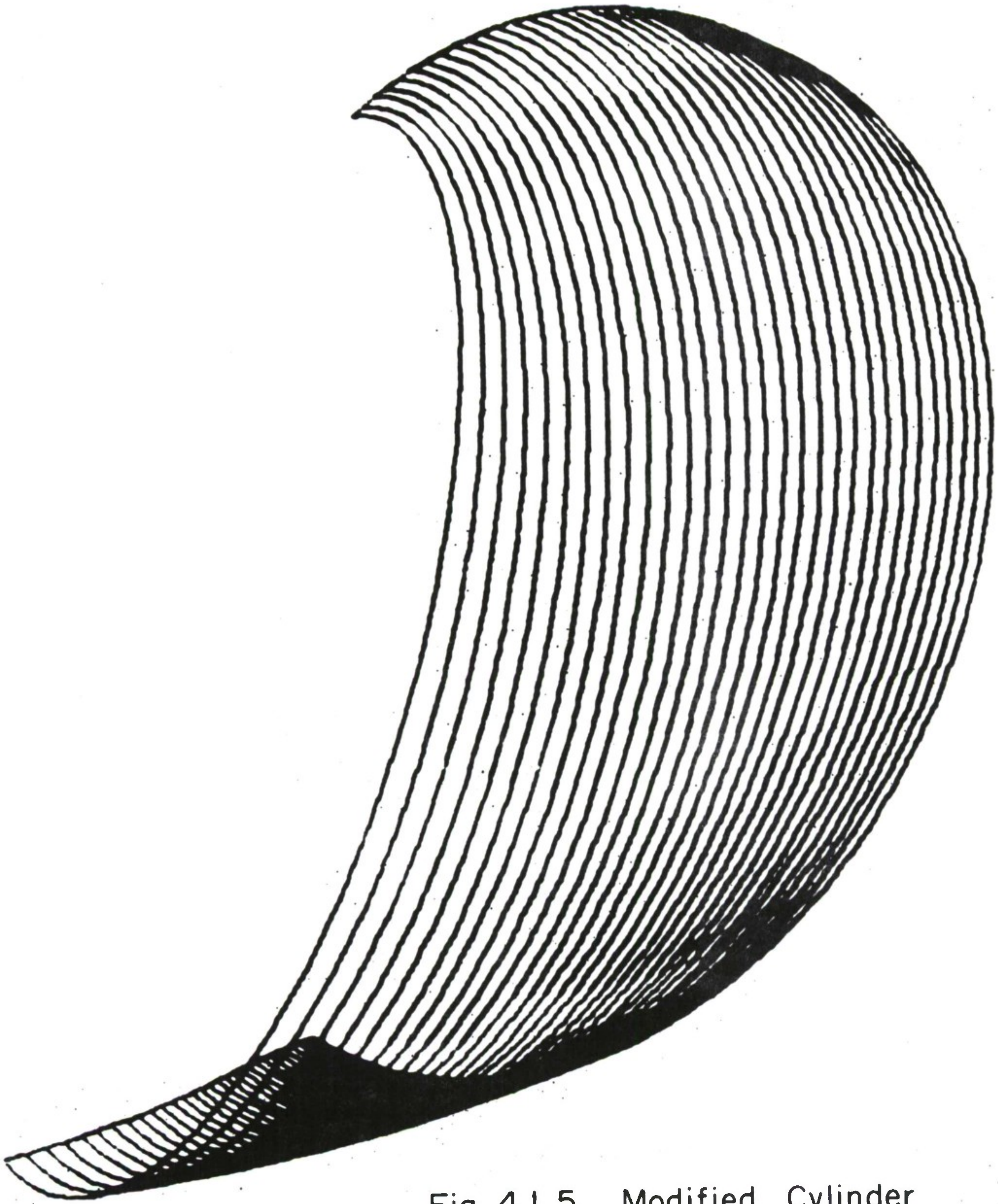


Fig. 4.1.5 Modified Cylinder

Two concerns were foremost in the design and coding of the routines for making the film. The first was to incorporate the theory and algorithms into a moderately flexible system in which as much responsive interaction as possible could be used to generate an aesthetic sequence of continuously moving views of the surfaces. Opportunity for experimentation with the composition and form was to be provided on the computer long before the sequence was committed to film. The second concern was to make the resulting program run as fast as possible during the crucial phase of generating a visual image of the surface. This was accomplished by using few subroutines in the critical areas, writing in machine language, and by unwinding loops into straight-line code. Subscripted arrays had to be removed since the PDP-1 has no index registers and must compute variable subscripts with the standard arithmetic instructions, using indirect addressing for references. As a compromise forced by the relatively slow speed of the computer and displays, magnetic tape was used to store some intermediate information.

The surfaces considered in the system are time-varying surfaces of the form

$$S_{ijk} = \alpha S'_{ijk} + (1-\alpha)S''_{ijk}$$

where the "mixing" parameter α, $0 \leq \alpha \leq 1$, is continuously adjustable from a potentiometer. Thus we can have a surface which changes from a plane to a hemisphere in the process of continuous deformation. Very interesting intermediate surfaces are formed in such cases as transforming a hemisphere into a quarter-torus.

The operation of the system is shown in Figure 4.2.1 and described as follows. The tensors for a set of possible surface patches had been computed by hand very early in this research and were now made known to the system. Compound surfaces formed from pairs of patches are defined as the objects of interest. Several such compound surfaces can be displayed at once, allowing a sphere to be formed from two hemispheres or reference planes to be added. The compound surface also contains a specification of the appearance it is to take on

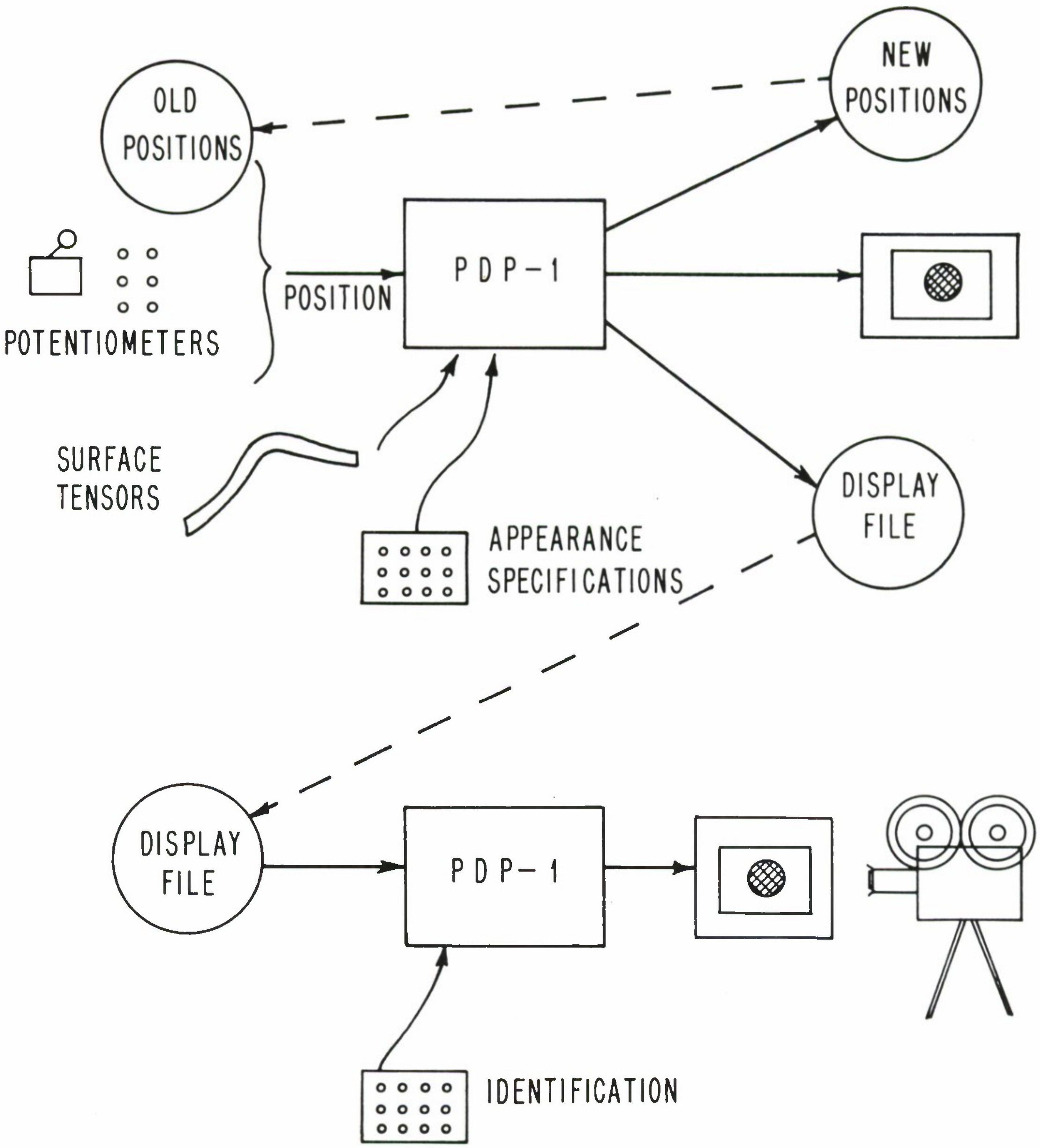


FIG. 4.2.1 FILM MAKING SYSTEM

the display, that is, the number of parametric curves and the number of points on the curves. For initial composition, a simple visual representation -- say three curves for each parameter, five points for each curve -- is displayed. Potentiometers are used to control the rates of rotation of the surface about the three different axes and to control the rate at which the compound surfaces metamorphose from one case to another. Translation along the three axes is possible by a set of switch controls, but this type of motion did not add much to the film sequence. During this continuous motion the successive positions in time -- represented by a 4 X 4 homogeneous matrix -- and the mixing parameter are recorded on tape. These computations for a simple representation can be performed at a rate of about five to ten frames per second, sufficiently close to real-time to make the potentiometer controls adequate for composing the film sequence. The visual representation can then be changed to a more suitable detail, say 17 curves and 33 points per curve. The position tape is used as data to generate the actual frames of the film for this detailed representation, which are recorded on another tape. This computation is considerably slower, on the order of one to five seconds per frame, depending on the number of surfaces displayed at once and the fineness of the visual representation. The perspective viewpoint is chosen at the same time as the visual representation and can be varied for different instances of the same sequence of positions. The final visual tape can be read at the maximum speed allowed by the tape drives and the display processor -- about ten frames per second -- to let the detailed sequence be viewed (but not modified) in almost real-time without recording on film. The images on the tape are then photographed a frame at a time on a high-quality 35-mm animation camera.

As is probably true in many interactive systems, the need to achieve fast response forced much time to be spent on the rather unilluminating problems of writing efficient machine code rather than on the more important areas of human-engineering and overall aesthetic design. These problems were of course aggravated by the nature of the objects being computed -- realistic representations of three-dimensional surfaces. It is clear that computer graphics is one major area of computational endeavor in which even the simplest of problems can be expected to reach the bounds of the available computing resources. It is also true that relatively minor improvements can be expected to have major effects on the practicality of a proposed procedure. In the case of these bi-cubic rational surfaces, either a special purpose curve generator as described above in subsection 2.2 or a conventional general purpose processor with a few hundred words of fast memory to hold the subroutines for computing a curve (including the homogeneous coordinate division and generation of display processor code) would enable a useful real-time system for the drawing of space forms using the representation discussed here. Even more simply, a highly optimizing compiler for an algebraic language with a library of standard matrix routines would have reduced the programming time for a substantial part of the movie system to a matter of hours, rather than the month or so it took. If the compiler had facilities such as the proposed I.B.M. PL/1 compile-time macros, it would be possible to expand the inner loops and achieve as fast running a program as could be expected, even on a computer like the PDP-1 with a simple fixed-point instruction set.

Note: Copies of a twelve-minute, 16-mm, black and white silent film made by this process may be obtained by purchase order to

Film Service Laboratory
62 Berkely Street
Boston, Massachusetts 02116
Telephone: (617)-542-1238

In ordering please request a reversal print of "Rational Bi-Cubic Surfaces" by Theodore M. P. Lee, Harvard University. The price will be about $40. The film includes examples of all the surfaces in Appendix I, each surface slowly changing into another surface. The film is to be projected at the standard sound speed of 24 frames per second. (Because of idiosyncracies in the 35 mm to 16 mm reduction process, a non-reversal print gives an annoying non-symmetric white border around the frame -- the reversal print will have black lines on a white background and will have higher contrast and definition than the non-reversal print.)

SECTION V

PROBLEMS FOR FURTHER RESEARCH

5.1 PROBLEMS FOR FURTHER RESEARCH

The following paragraphs outline a few of the important questions open for future research. I cover here only the more mathematical questions; clearly the problems associated with organizing these results into an effective, human-engineered system for computer aided design must be answered at some time.

Since a boundary curve can be degenerate, patches with only two or three sides are possible. Such patches by virtue of the degenerate boundary curve must be treated differently as far as tangent plane continuity is concerned. Exactly what constraints can be imposed and what types of surfaces with degenerate curves exist are open questions.

Another question of continuity constraint is the general problem of two overlapping surfaces, say an airplane wing and fuselage. Here we are attempting to ask for surface continuity in the middle of a patch, not on the boundary. Of course, we could use contours in the interior of a patch to define a new subpatch at the boundaries of which continuity is to be imposed. But this may not always be possible, nor may it be the most general solution.

Further results are needed in specifying surfaces from external data. Personally, I prefer the surface-molding approach in which on-line feedback is continuously present as contrasted to the point surface-fitting approach currently used by most surface design systems. However, there are many occasions in which it would be expedient to use such easily displayed surfaces as these to approximate a surface described in some other form, perhaps from measured coordinates.

Even using these bi-cubic rational surfaces as molded surfaces, there are not at present sufficient techniques for manipulating the display. For instance, efficient methods of performing small variations in the surface are obviously needed.

I will do no more than mention the question of discovering hidden lines in a figure composed of these surfaces. The problem for this geometry is obviously very difficult and defies a simple analytical solution. Even the application of Warnock's algorithm [23], effective as it is, may not be the answer, since his techniques have been found very susceptible to Mach band distortion on curved surfaces approximated by planar sections. Such distortion gives the surface a "fluted" appearance at the joints of the planes caused by a psychological enhancement of the intensity discontinuity.

APPENDIX I

EQUATIONS OF TYPICAL OBJECTS

We list here the equations of six typical surfaces. They were used for the photographs in Section IV and as the basic surfaces used in the movie described in subsection 4.2. The sphere, cone, cylinder and bottle describe one-half of the surface by that name; the torus describes one-quarter of a complete torus. The bottle surface was formed as a surface of revolution whose generator is the function F_0; it looks vaguely like a bottle neck or wine glass stem.

1. Cylinder

$hx = 2v - 1$

$hy = -2v^2 + 2v$

$hz = u(2v^2-2v+1)$

$h = 2v^2 - 2v + 1$

2. Cone

$hx = (2u-1)v$

$hy = (-2u^2+2u)v$

$hz = v(2u^2-2u+1)$

$h = 2u^2 - 2u + 1$

3. Saddle

$hx = 2u - 1$

$hy = 2v - 1$

$hz = 4u^2 - 4u - 4v^2 + 4v$

$h = 1$

4. Sphere

$$hx = (2u-1)(-2v^2+2v)$$

$$hy = (-2u^2+2u)(-2v^2+2v)$$

$$hz = (2u^2-2u+1)(2v-1)$$

$$h = (2u^2-2u+1)(2v^2-2v+1)$$

5. Torus

$$hx = (36u^2 - 36u + 9)(6v - 3)$$

$$hy = (36u^2 - 36u + 9)(18v^2 - 18v + 4)$$

$$hz = (6u - 3)(18v^2 - 18v + 5)$$

$$h = (18u^2 - 18u + 5)(18v^2 - 18v + 5)$$

6. Bottle

$$hx = (2u - 1)(2v^3 - 3v^2 + 1)$$

$$hy = -(-2u^2 + 2u)(2v^3 - 3v^2 + 1)$$

$$hz = 2(2u^2 - 2u + 1)(v - \frac{1}{2})$$

$$h = 2(2u^2 - 2u + 1)$$

APPENDIX II

PROPOSAL FOR A DATA STRUCTURE

The mathematics of Sections II and III provides a set of tools to be used in the specification and display of curves and surfaces. These results provide the muscles for a computer-aided design system but neither the skeletal framework holding it together nor the nervous system enabling and controlling action; neither do they indicate the ways in which the system is to relate to the world, nor the means and direction of its evolution and growth. In this appendix we propose a data base structured so as to serve as a framework into which the mathematics will naturally fit and onto which it will operate. This data base is to contain explicit models of the connections between the concepts involved in talking about surfaces and is designed to expand qualitatively as these concepts are expanded.

This material was developed in response to the difficulties encountered in programming the examples described in Section IV. It became clear that a significantly useful implementation of the curve and surface theory would require the development of routines that really "knew" about surfaces rather than ones that were only capable of blindly manipulating the tensors in the fashion of the present programs. The primary evidence for this conclusion was the awareness that throughout the development and use of the programs, experience would suggest continual changes in their mathematical and functional appearance. These changes need to be incorporated into the system in as painless a manner as possible. A tentative solution to the problem was suggested by the extreme generality and flexibility of Feldman's associative data structure [8]. Since none of the material in this section has been tested, it is presented only as groundwork that should not have to be retraced in the future.

The intent of this material is to demonstrate the necessity, practicality and utility of a highly general data structure in the particular context of computer-aided design of space forms. We will pursue the details only so far as is necessary to make these points and to illustrate the possibilities inherent in the homogeneous tensor representation chosen for curves and surfaces. In this presentation there will be many design questions -- such as the manner in which matrices are to be represented -- whose resolution will depend upon experiment. In these areas I am being purposely vague in order not to make an unfounded commitment to some particular implementation. I make no claim of originality for the immediately following general comments on data structures, although I can cite no particular source for them; their sole purpose is to set the stage for the case under discussion.

Philosophically speaking, the purpose and appropriateness of a data structure are operationally determined by the uses to which it is put, that is, by the questions the data structure is expected to help answer and what those answers are expected to accomplish. We think of the data structure as being used to form a model of some possibly changing set of concepts; instead of asking questions about the concepts directly, we ask questions about the model and expect that the answers can be interpreted as answers to equivalent questions asked about the concepts.

The choice of a data structure thus implies not only a particular modelling of the concepts but also a modelling of the questions: to answer a question put to the set of concepts, we map the question to its appropriate model, ask <u>that</u> question or questions of the data base, built in accordance with the chosen data structure, and map the answer back to an answer of the original question. If we choose a simple data structure to model a complex set of concepts, a simple question asked in the language of the set of concepts will be mapped into a complex question asked in the language of the data structure. If either the resulting question or the mapping determining it -- presumably some program -- is too complex, it is obvious that the data structure is unsuited to finding an answer to the original question; in fact, it may be impossible to do so. In choosing a data structure, if we do not know in advance all the questions to be asked, we at least ought to be able to convince ourselves that the types of questions likely to arise will not be too difficult, the difficulty being measured in terms of the mappings necessary to apply the question to the structured data base.

Now let us talk about curves and surfaces. The claimed result of this dissertation is that the homogeneous tensor representation of surfaces as rational bi-cubics is desirable. We thus assume that part of the representation of a surface will be such a tensor. Immediately we notice that this tensor is not unique, since there are several different such representations suitable for different purposes:

1. The polynomial representation is good as an abstract prototype of the surface.
2. The endpoint derivative representation is good for specifying the curve from geometrical considerations.
3. Any of several difference equation representations is good for generating a visual representation of the surface, the particular difference equation used dependent on how many and what type of curves we wish to display.
4. The code driving the display is itself a representation, making the surface visible.

In addition to these explicit representations applicable to the general bi-cubic rational surface, we have the representation of a product surface in terms of two curves, each curve in turn represented as a matrix. And then we have a surface of revolution represented as a product surface in which one of the curves is an arc of a circle and the other an arbitrary plane curve. Or, by the results of subsection 3.8, we can represent a surface by six curves and a few additional parameters. In the course of working with a particular surface, all of these different representations might be useful and might exist at the same time. We can also conceive of additional representations or characterizations as yet undeveloped in detail, such as requiring that a surface be a torus, a sphere, the hull of a three-masted schooner, or the right tail-fin of an Edsel station wagon, assuming all such concepts can be formulated to a sufficient degree of exactness. We will not succeed in building an interesting surface manipulation system unless it is capable of freely wandering about this proliferation of representations, choosing the appropriate one for the task at hand.

If we grant that any or all of these various representations of a surface need to be modelled, we see two types of questions to be answerable by use of the data base. Prototypes of these questions are:

1. "What is the display code to be used to draw this surface?"

and

2. "I find that there is no display code for this surface; how do I generate it in order that I may answer question 1?"

The type of answer to the first question is clear; to the second question we would expect an answer of the form: "If you have a finite difference representation of the surface, you can use subroutine GENERATESURFACE to compute the display code." Notice that this answer implies the additional question -- "What is the finite difference representation of the surface?" -- which may lead to further questions about finite difference representations. In the use of most common data structures, it is only questions of type (1) that are able to be answered directly; questions of type (2) are treated implicitly by the programs that massage the data base. Changing these programs in order to expand this set of answerable questions is usually very painful.

Another way to characterize the two questions is to notice that in the first, explicit information answers the question; in the second, explicit information answering the question about finite difference representations yields the implicit answer to the question about display code -- that is, implicit in the fact that the object has a finite difference representation is the fact that display code for that object can be generated by a particular subroutine.

If we permit the data structure to allow modelling of the type of information implied in question (2), we can provide an answer to newly arising questions in terms of previously known questions. For instance, if we define a sphere as a particular kind of surface, describing the way to obtain a tensor representation of the surface from some particular simple representation of the sphere, we can answer any question about the sphere if the same question can be answered about a general surface. Continuing, if we now decide that it is useful to talk about the boundary curves of a surface, specifying how that is to be done in terms of the tensor representation of a general surface, we can ask for the boundary curves of a sphere, even though the sphere is initially represented only by its center and radius, having no explicitly represented boundary curves.

Abstracted from the above, the types of information that need direct representation are as follows:

1. Explicit pieces of numeric information, such as the contents of a tensor or matrix.

2. Objects in the abstract, such as a particular surface or curve, apart from any specific numeric characterization.

3. Relations between objects, such as "this tensor is the polynomial representation of that surface," or "this curve is a boundary curve of that surface."

4. Generic information, that is, the answers to questions of type (2). We will suggest a way in which for this problem area this type of information can be modelled in terms of the first three.

Categories (1) through (3) are to be structured to allow direct answers to questions of type (1) -- given an object, any of the pertinent properties of the object are intended to be retrieved by examining the appropriate relations holding between the object and the desired property. (This will be made clearer shortly.) Any of a number of schemes for representing relations could be chosen so as to permit the direct access to this kind of information. We will briefly mention later one such scheme, but our chief concern is to point out in this context the utility of the general concept of representing information in terms of relations.

To amplify category (3) recall [9, p. 49] the formal definition of a relation as a set of ordered pairs drawn from some universe, such as $\leqq = \{\langle x, y\rangle | x \leqq y\}$. An <u>instance</u> of a relation is just one of the ordered pairs -- $\langle 2, 3\rangle$ -- which is better expressed as the ordered triple $(2 \leqq 3)$ in order to emphasize the relation of interest. Typically, we will be concerned with mathematical relations such as set membership or with practical relations such as the one holding between a surface and any of its several representations. Both an entire relation and a specific instance of a relation must be considered as objects in themselves, which can participate in relations with other objects. In this way it would be possible to indicate, for instance, that the relation $\leqq$ is transitive or that the particular statement "The finite difference representation of the surface A is B" was derived by using generic information about finite difference representations.

If we thus talk explicitly about relations and represent them explicitly in the data base, it becomes possible to introduce new concepts -- expressed as relations -- and to relate those new concepts to existing concepts in an operationally useful manner. This is, of course, not a new idea; I present it here with the claim that the problems here discussed are well-suited to solution by these means.

At this point it seems best to illustrate what I mean by an extended set of examples. These examples are contained in the graphs in Figures A.2.1 through A.2.12 and will be discussed in detail in the next several pages. The graphs are directed, with labels (possibly empty when unessential) on both nodes and arcs. The graphs indicate some portion of the information to be represented in the data base and thus consist of a number of relations between objects, a relation being defined by the explicit representation of its instances. The label on an arc or node is intended to be a unique name for the object represented by the arc or node. The instance (A R B) of a relation R is indicated by making an arc labelled R from a node labelled A to a node labelled B. The fact that an arc and a node may have the same name is a result of allowing relations to be treated as objects; in order to treat an instance of a relation as an object, we will circle the label appearing on the arc representing the instance and treat that circle as if it were a node in the graph.

As our first example, let us look at the information intended to be retrieved by the prototype questions (1) and (2) above. If we denote the particular surface concerned by S, the first question translates to the request "Find x such that the relation (S DISPLAY-CODE x) holds."

The answer expected from this question would be found in (S DISPLAY-CODE D) as the object D. The second question can be rephrased, not quite so formally, as "Is there a statement in the data base which could be used to deduce the existence of a subroutine to compute display code for an object, and if so, can that deduction be made for the particular object S?" The answer to this question would be supplied by a representation of the statement "For all surfaces x the existence of a finite difference representation y implies the existence of display code z whose value is computed from y by the subroutine GENERATEDISPLAY."

Figures A.2.1 and A.2.2 are two models for this statement; an exegesis of the models follows. To begin, the statement can be written formally as the conditional proposition

$(\forall x, y)$ ($x \varepsilon$ SURFACES) (x FIN. DIF. REP. y) $\Rightarrow$

$(\exists t, z)$ (x DISPLAY-CODE z) (t ARG. y) (t SUBR. GENERATEDISPLAY) ($t \rightarrow z$)

where the object t represents the application of the subroutine, the relation ARG. describes y as an argument of the subroutine and the relation SUBR. gives the name of the subroutine to be called; the instance ($t \rightarrow z$) says that this subroutine call will give the value of z.

Figure A.2.1 represents the statement by labelling the four nodes corresponding to the quantified variables x, y, z, t with the appropriate quantification symbol; the labels for the variables are omitted since they are irrelevant and only serve to distinguish the separate variables, which is done here by using separate nodes. The two relations serving as the antecedents of the conditional statement are drawn with heavy lines; the four consequents are drawn with dotted lines. In general, we read a diagram such as this by saying "if all the relations indicated by heavy lines are true, for appropriate substitutions for the quantified variables, then we can deduce that all the relations indicated by dotted lines are true."

To actually represent this statement strictly in terms of objects and relations, we would have to use a scheme like that in Figure A.2.2.

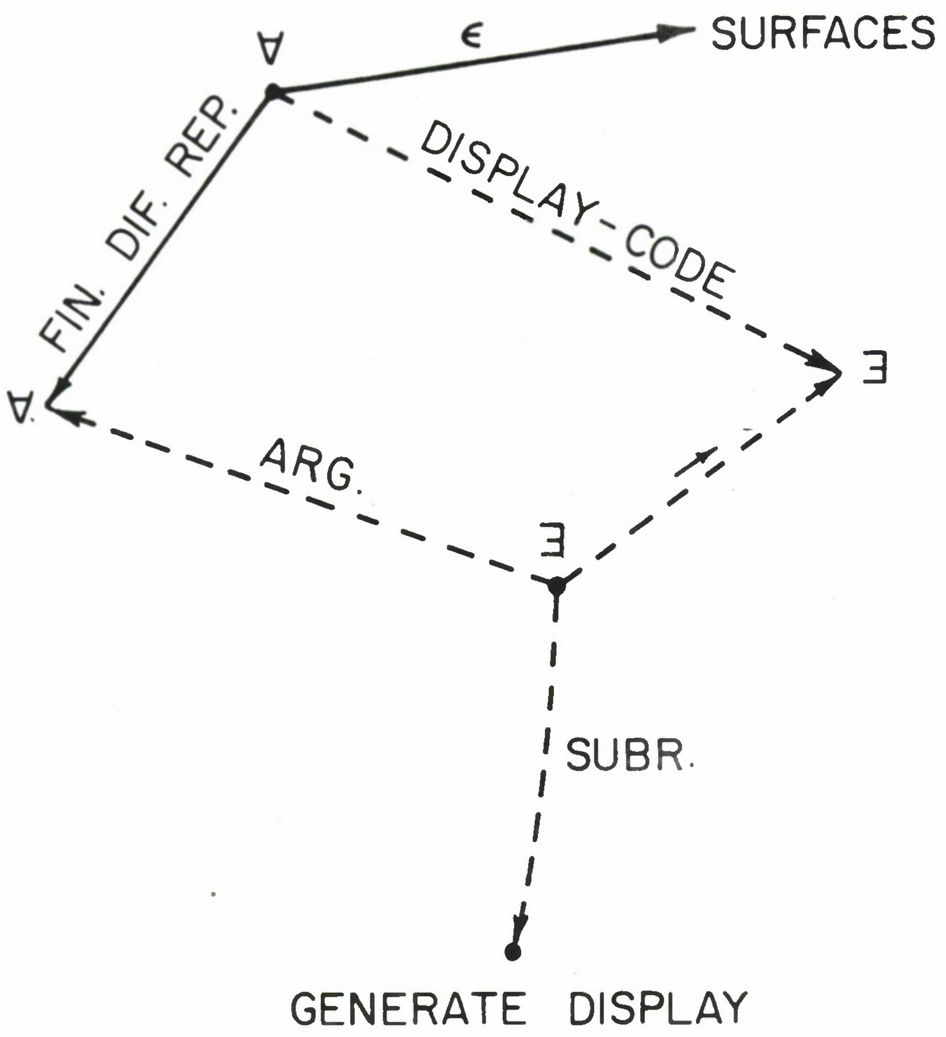


FIG. A.2.1 REPRESENTATION OF DISPLAY CODE DEFINITION (SIMPLIFIED)

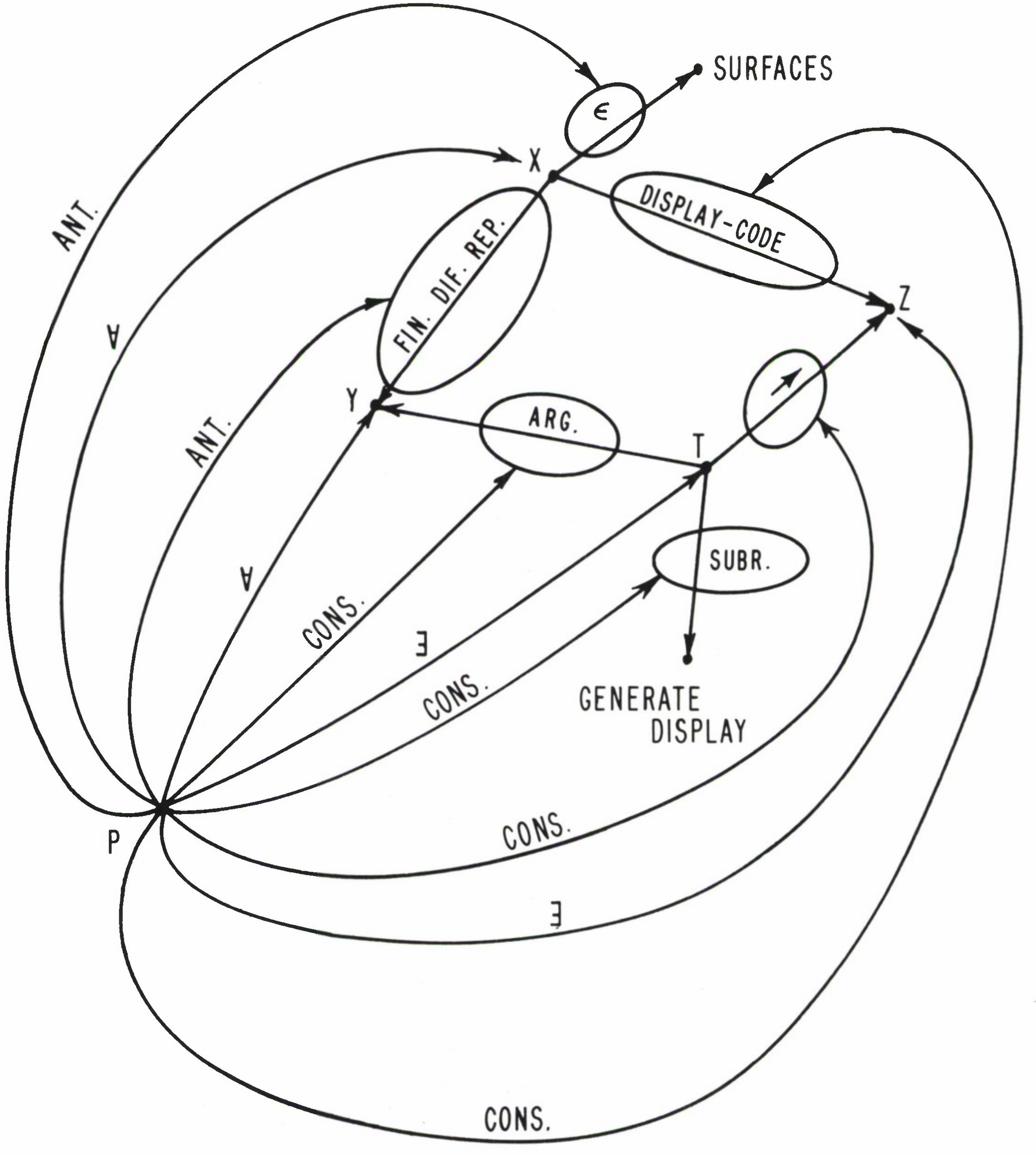


FIG. A. 2.2 REPRESENTATION OF DISPLAY CODE DEFINITION (DETAILED)

The node P groups the set of relations and variables in the statement together; the part played by each object or instance of a relation in the statement is indicated by relating that object or instance to P. Quantified variables are related by the quantification symbol, consequents by the relation CONS, and antecedents by ANT.

We would apply this representation to answering question (2) by translating question (2) into a sequence of questions of type (1), namely,

(a) "Find existentially quantified variables x, t and a universally quantified variable z such that the instances (x DISPLAY-CODE z) and $(t \rightarrow z)$ are consequents of some statement P."

(b) "For every antecedent of the statement P, verify that the antecedent relations hold when the universally quantified variables are replaced by the appropriate particular objects in the data base, such as by substituting the particular surface S for the variable x. If all the antecedent relations hold, create new objects for the existentially quantified variables and enter them into the data base via the consequent relations."

We will not explore this deductive process much further; the literature on deductive question-answering systems [3, 25, and works there cited] seems to indicate the feasibility of this approach and covers in much more detail than I care to know about at the present the precise algorithms to follow. I claim without other than empirical justification that the form of the statement represented by these figures is sufficient to answer the questions of type (2) that arise in this application. The subsequent examples are all of this type -- conditional expressions indicating what relations can be deduced from what relations -- and in a few pages cover a good cross section of the concepts developed in the previous sections.

(As an aside, the fact that the types of deduction to be used in accessing the proposed data base can be limited to this relatively simple form ought to point heavily toward the likelihood of a practical system based on these ideas.)

Before illustrating the representation of some specific concepts about surfaces, we introduce representations of three mathematical concepts that will simplify the subsequent representation of the surface material. The concepts to be discussed are

1. set membership
2. change of basis transformations, and
3. matrix multiplication.

Representing these concepts directly in the data base rather than merely using them in programs serves both the purpose of forcing the concepts out in the open where they will be subject to careful definition and the purpose of enabling their application to a wide variety of circumstances.

In Figure A.2.3 is represented the statement that a member of a set is a member of any including set; that is, for any objects x, y and z, given $(x \varepsilon y)$ and $(y \subset z)$, we can deduce $(x \varepsilon z)$. Having introduced this concept we no longer need to completely characterize objects which are defined as belonging to subclasses of existing classes; for instance, having defined the class of spheres to be a subset of the class of surfaces, any deductions applicable to an object S bearing the explicit relation (S ε SURFACES) will be applicable to an object T bearing the relation (T ε SPHERES), assuming any other antecedent conditions also hold.

In Figure A.2.4 is defined once and for all the inclusion relationships among the classes of interest. Having done so we do not need to define all the properties of each class of objects and can concentrate on the essential properties peculiar to that class.

The essential differences between the various tensor representations of surfaces can be phrased in terms of their different functional bases. As mathematicians we know how to transform a tensor represented in one basis to a tensor represented in a different basis; namely, if there is a transformation between two bases, a representation of an object in the range basis is the matrix product of the representation in the domain basis and the transformation matrix. Figure A.2.5 is a diagrammatic version of this statement -- its precise interpretation would be that the indicated matrix product relates the two representations.

$$\begin{tikzcd}\forall \arrow[r, "\in"] \arrow[rr, dashed, bend right, "\in"'] & \forall \arrow[r, "\subset"] & \forall\end{tikzcd}$$

FIG. A. 2. 3 SET MEMBERSHIP

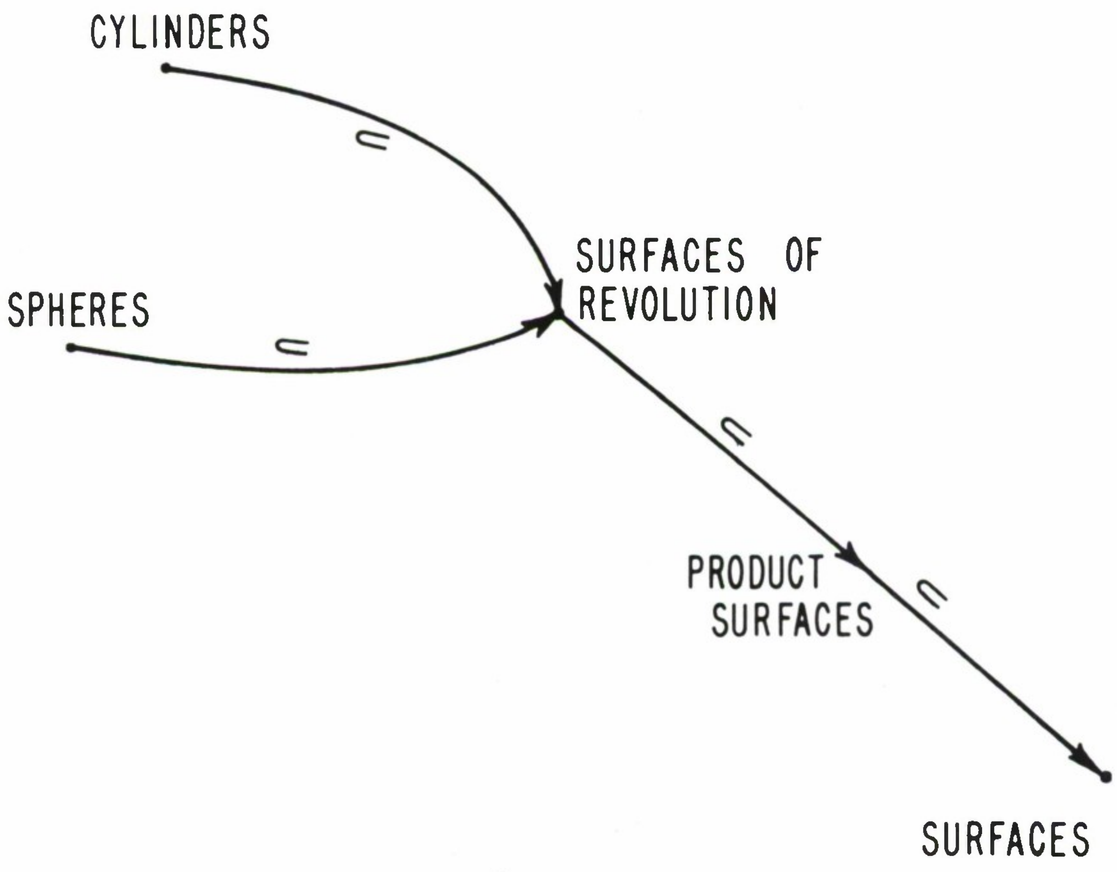


FIG. A. 2. 4 HIERARCHY OF OBJECTS

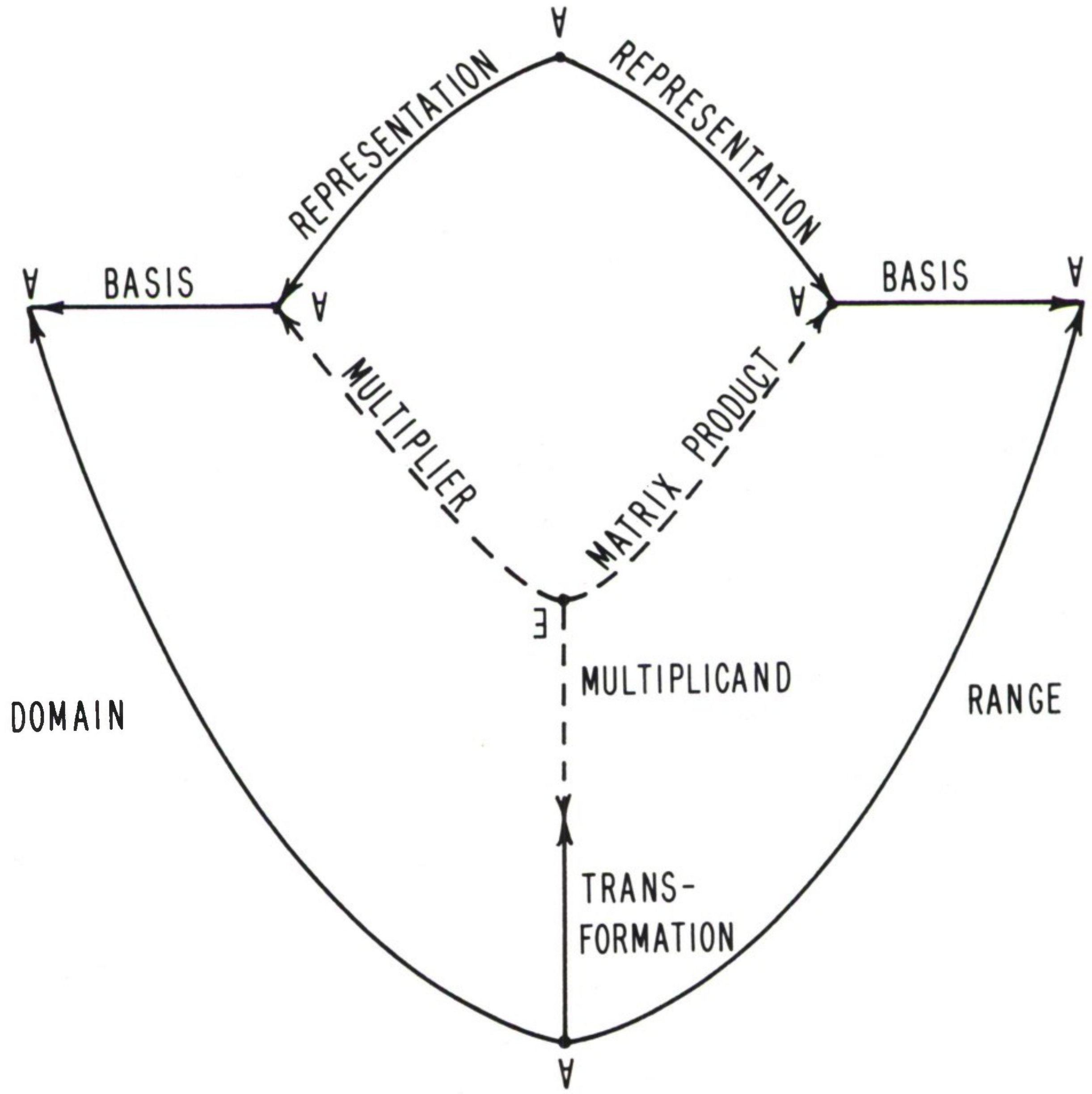


FIG. A.2.5 TRANSFORMATION OF BASIS

Hereafter, to enable the conversion of a representation of a surface from one basis to another, we only need define the particular transformation matrix relating the bases.

In the above paragraph we have defined the change of basis transformation in terms of a matrix product, representing the matrix product symbolically. In Figure A.2.6 we represent the additional fact that some subroutine -- here labelled MATMUL -- will compute the value of the product of two matrices. Separating the statement that one matrix is the matrix product of two other matrices from a definition of matrix product in terms of a subroutine serves to separate the computational problem of determining a value from the operational problem of deciding what to do; in particular, if we never ask for the value of some matrix which is the product of two matrices, we never need to deduce the fact that the subroutine MATMUL will give that value.

The transformation from a polynomial basis $[u^3 u^2\, u\, 1]$ to the end-point derivative basis $[F_0 F_1 G_0 G_1]$ is given simply by the matrix we have called M and is represented by Figure A.2.7. The transformation from the polynomial basis to the particular basis needed to generate a surface by finite difference methods is a function of the number and direction of the curves to be displayed on the surface and the number of points on each curve. We assume a subroutine COMPUTEDISPLAYBASIS to compute the value of the matrix for this transformation, using the material of subsections 2.2 and 3.5. Figure A.2.8 illustrates these concepts and would be applied to give the finite difference representation of a particular surface once the particular family of parametric curves was chosen.

In a similar fashion we can use the proposed data structure to represent the algorithms to be used to build a surface specified by geometric conditions. Let us suppose the designer wishes to define a portion of a cylinder as a particular kind of surface. (We speak about a portion of a cylinder since we cannot draw a complete cylinder at one surface patch, this because we cannot draw a complete circle, but rather only any part less than a complete circle.) Granting for the moment that

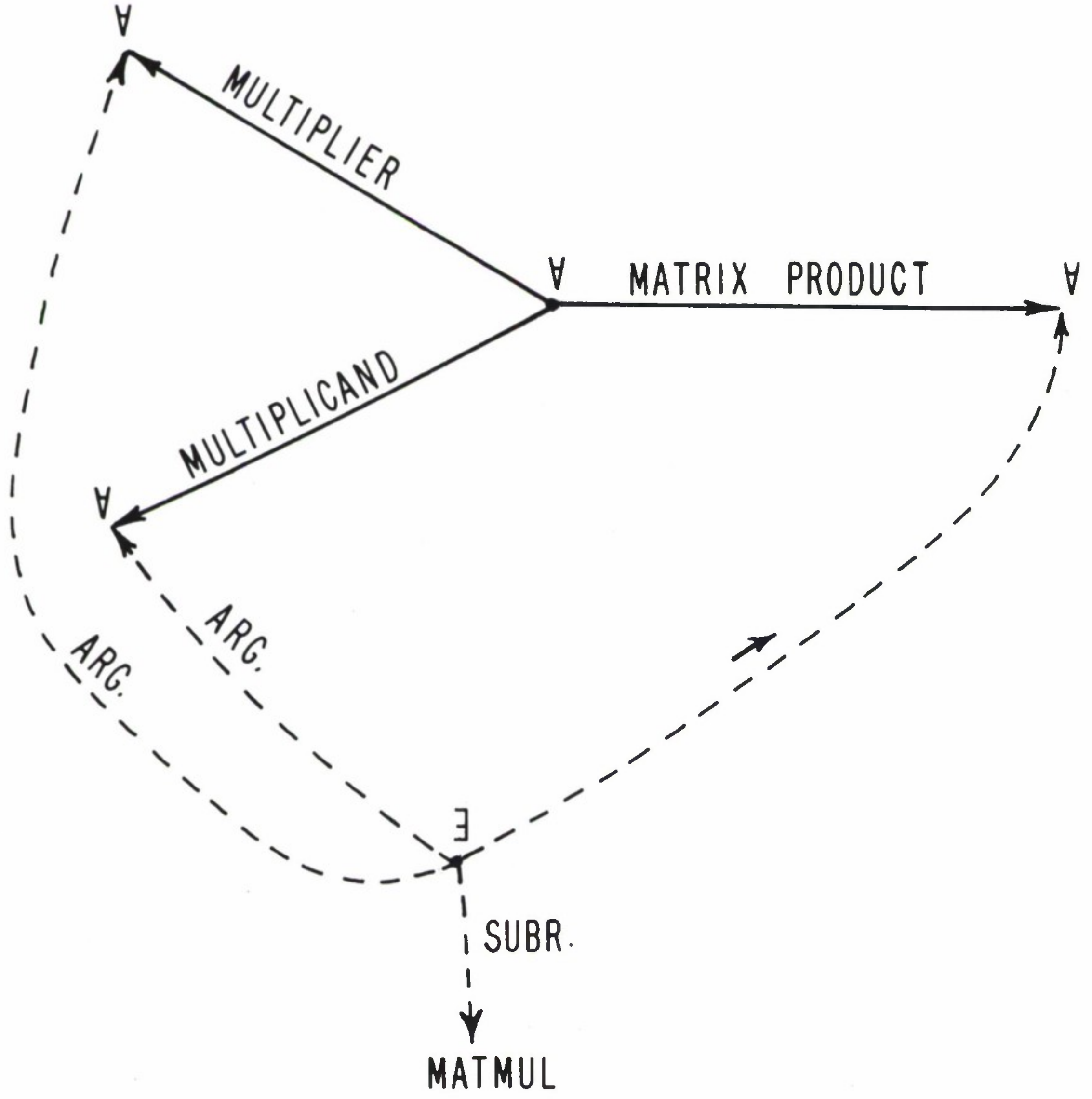


FIG. A.2.6 MATRIX PRODUCT

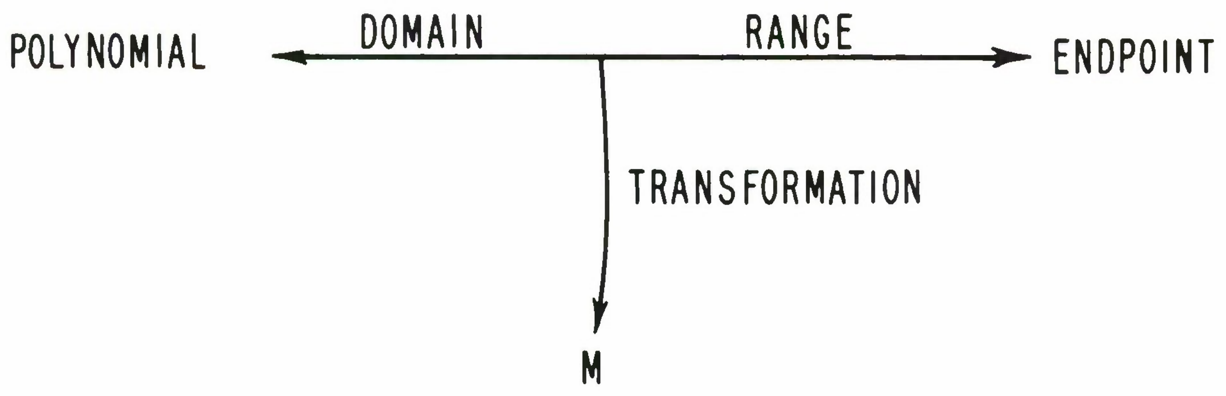


FIG. A.2.7 POLYNOMIAL / ENDPOINT TRANSFORMATION

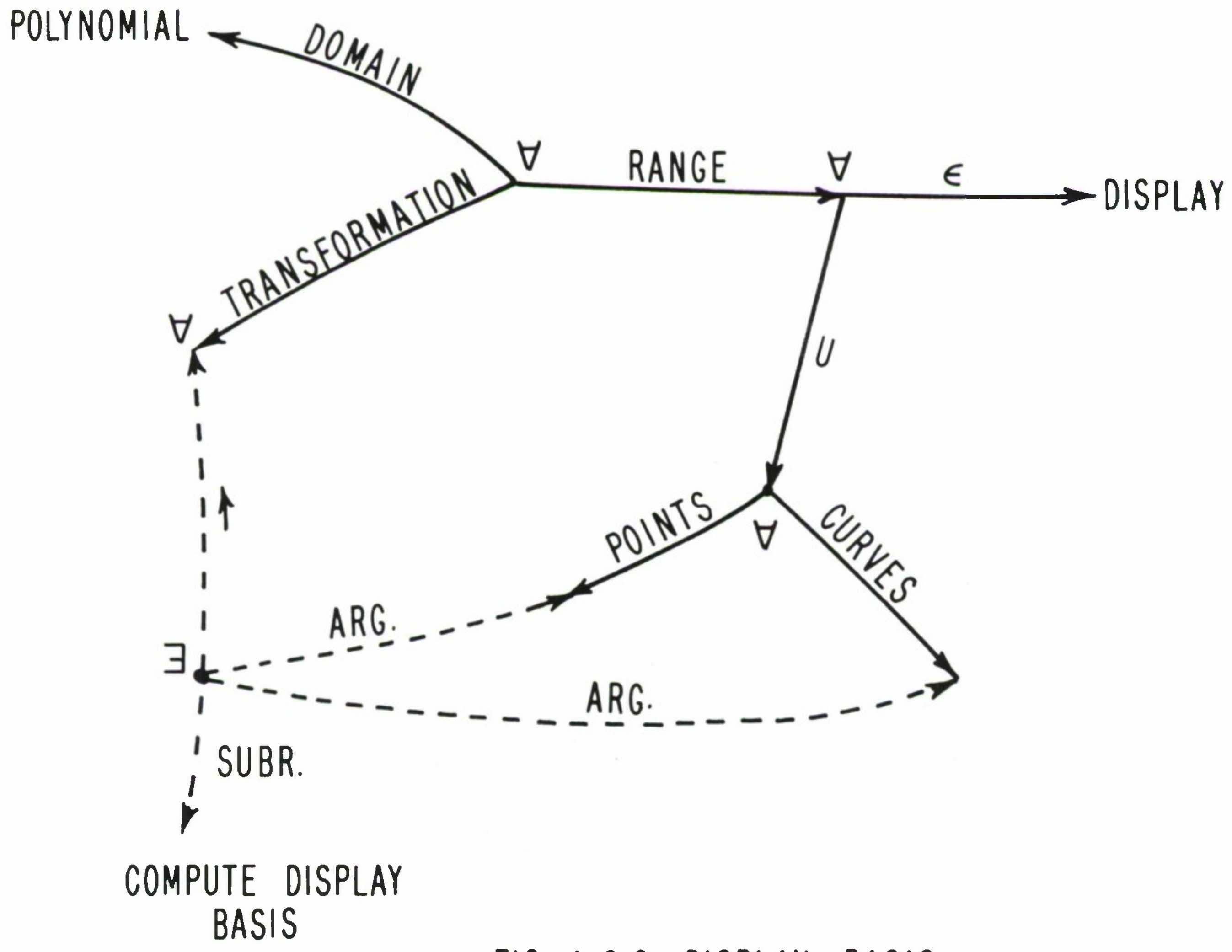


FIG. A.2.8 DISPLAY BASIS

such a definition can be given, Figure A.2.9 represents the particular instance GEORGE of a cylinder, including the fact that the cylinder is to be displayed using 5 curves in the u = constant family and 2 curves in the v = constant family, 17 points per curve in the former and 33 points per curve in the latter. The particular portion of a cylinder specified is to have a height of one and is to subtend an angle of 270°. In order to display this surface we need to go from the specification of this particular cylinder to a tensor representation in the polynomial basis; by the algorithm expressed in Figure A.2.8 we can then find the particular finite difference representation to draw the surface.

At this point we could take either of two approaches. The first would be to define a subroutine MAKECYL that would compute the tensor representation of a cylinder given the specified dimensions as arguments; this is represented in Figure A.2.10. The second approach would be to define the cylinder as a particular type of surface of revolution, define surface of revolution as a particular type of product surface, and then define a product surface as a particular type of general surface. This approach has the advantage of directly representing the distinguishing facts about the cylinder and would now allow, for instance, a sphere to be readily described as a different particular type of surface of revolution, with the rest of the machinery directly applicable to its generation.

We will not represent the characterization of the cylinder as a surface of revolution since that would depend on some unilluminating assumptions about the representations of circle arcs and straight lines; we will indicate how the rest of the machinery might be represented. Let us assume a surface of revolution to be specified by a generating curve and a particular arc of a circle; to call this a product surface, we need only relabel the generating curve and circle to correspond to the two curves characterizing a product surface as in subsection 3.9 -- we will call these GENERATOR-ONE and GENERATOR-TWO. This relabelling is indicated by the representation in Figure A.2.11. Then to obtain the proper general tensor representation of the product surface,

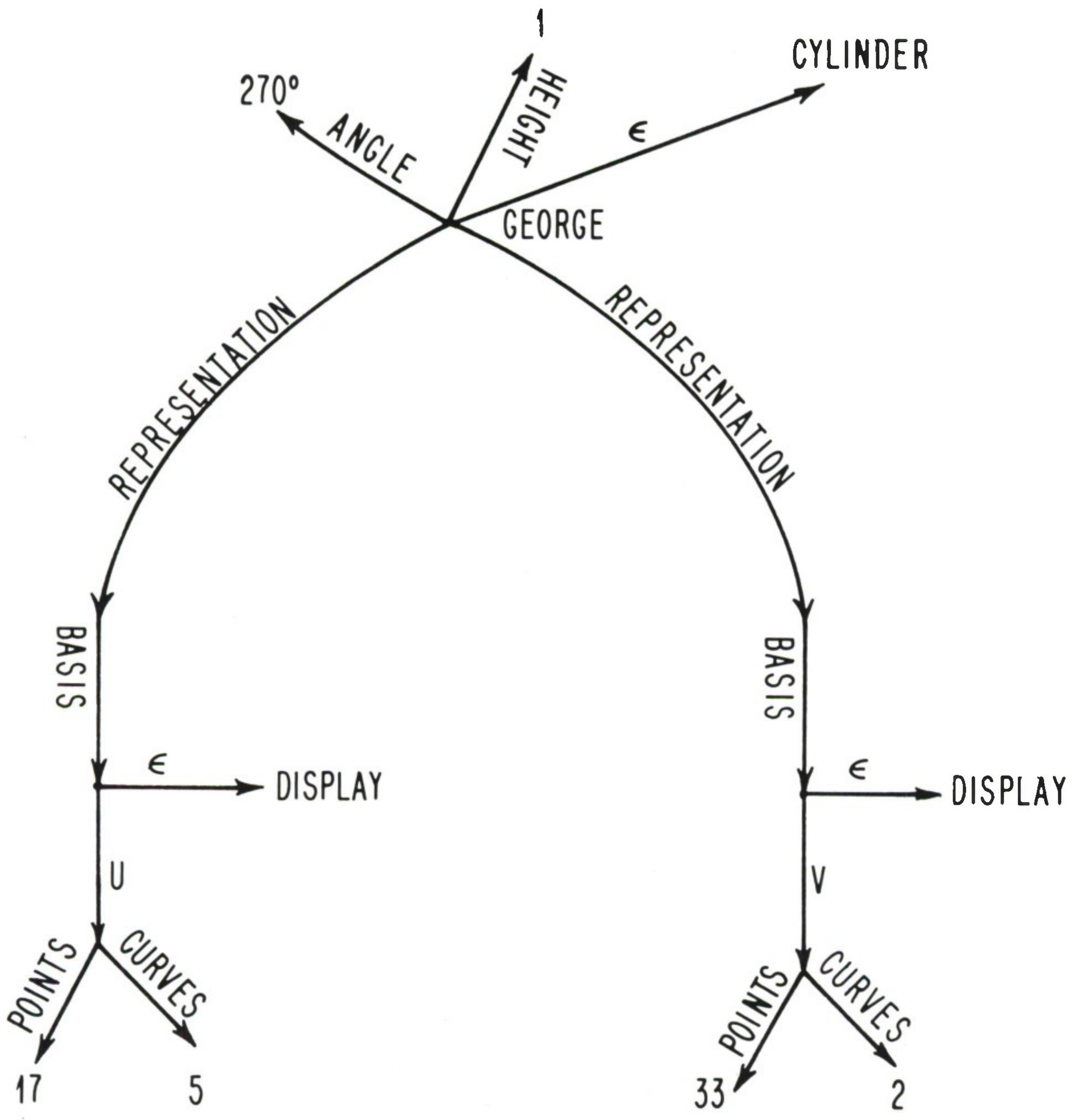


FIG. A.2.9 INSTANCE OF A CYLINDER

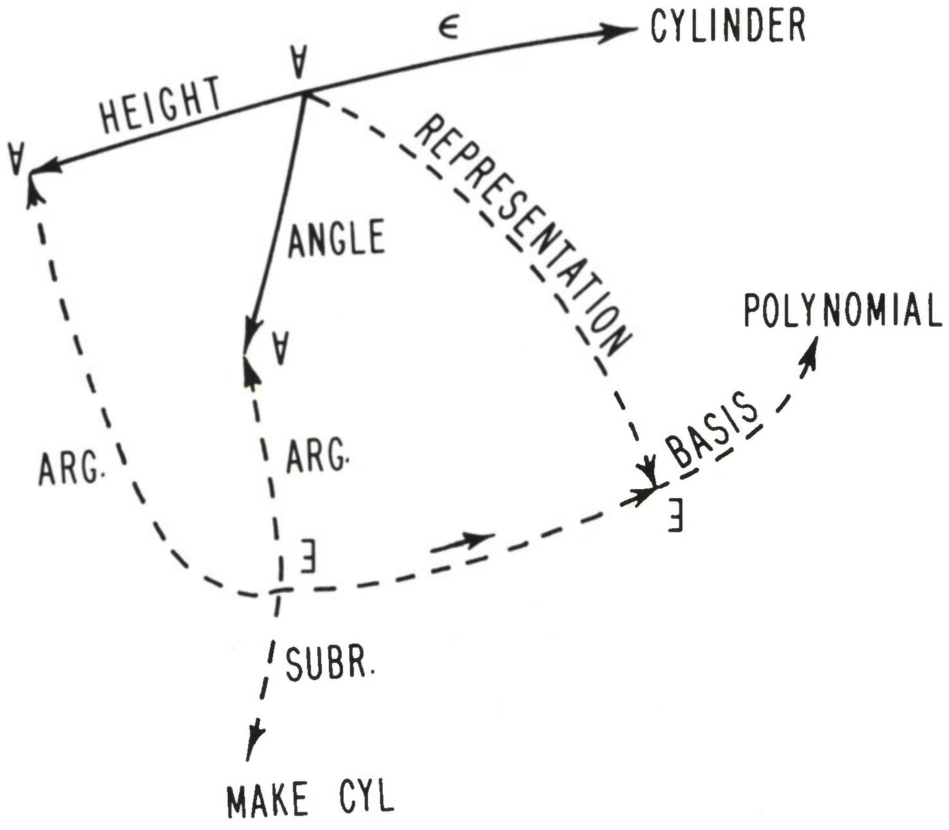


FIG. A. 2. 10 A CYLINDER DEFINITION

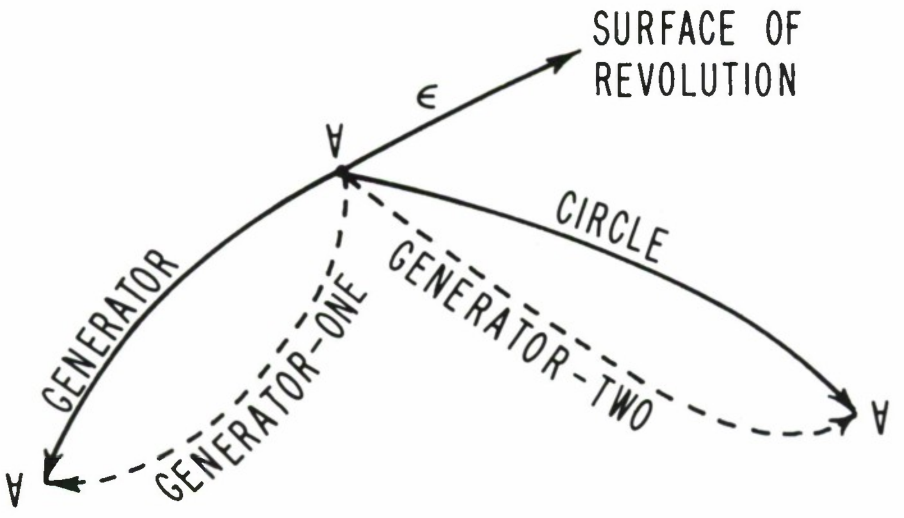


FIG. A.2.11 SURFACE OF REVOLUTION DEFINITION

we define a subroutine PRODSURF that multiplies the matrices representing the curves in the proper way to give the value of the tensor -- notice here that the functional basis of the tensor will be the same as the single basis of the representations of the two generating curves. These concepts are represented in Figure A.2.12, completing the set of deductions necessary to draw surfaces of revolution.

The examples just discussed are, of course, not complete nor completely accurate, even so far as they go. In particular, the change of basis transformations would be more accurately expressed in some tensor-like notation, since we have to take into account that we are dealing with two bases, the functional basis of the curve -- some particular set of polynomials -- and the basis of the four-dimensional projective space in which the object is represented. This latter basis is not mentioned at all in the examples, but must of course be included since it is what determines the position, orientation, and perspective view of the object. For instance, we would like to allow compound objects composed of some set of previously defined objects in some particular orientation. We would hope that a spatial transformation applied to such a compound object would be computed by applying it first to the transformations expressing the internal orientation of the individual objects in the compound object, and then applying that compound transformation to the representations of the internal objects. This would, of course, be a recursive process, since a component of a compound object might be a compound object itself.

An implementation in terms of this data structure will have to provide the following capabilities:

1. Representation of instances of relations (ordered triples).
2. Access to such triples using any one, two, or all three of the possible elements of the triple as addressing criteria, possibly producing multiple answers.
3. Representations of numeric or other values of objects.

The important problem to be solved in such an implementation is that all relations although written as one-way directed arcs are to be

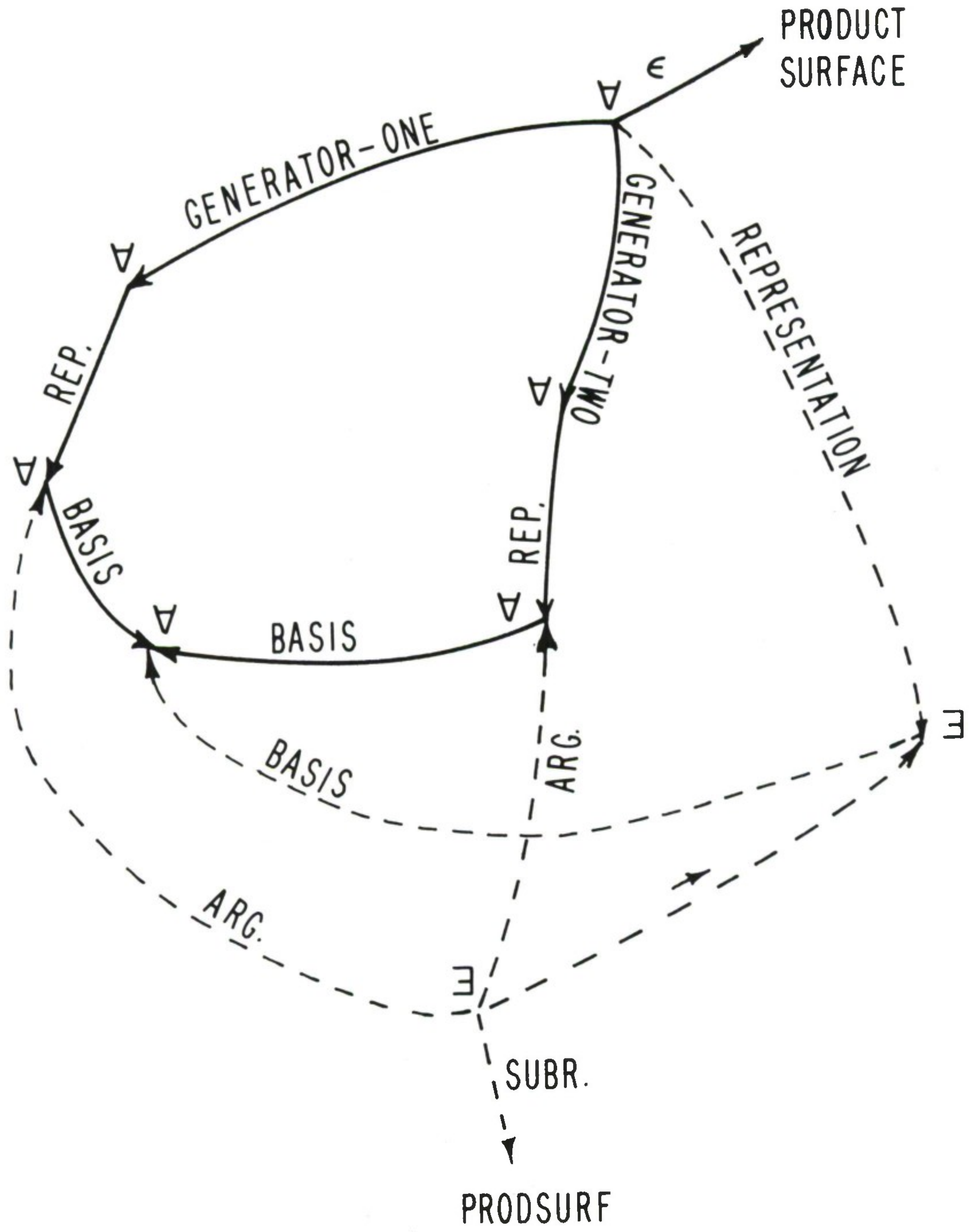


FIG. A.2.12 PRODUCT SURFACE DEFINITION

two-way arcs, in which access paths exist in both forward and backward directions. Most conventional data structures do not provide automatically for such bi-directional access paths in a symmetrical manner. Even fewer permit the access path itself to be a possible item of interest. The LEAP language and data structure [8, 18] on the TX-2 at the M. I. T. Lincoln Laboratory is an example of a fairly efficient way to represent this type of information. By using hashing techniques, it stores each instance of a relation in three ways in order that given any two of the components of a triple, there is a virtually direct path to the third component.

The particular class of problems arising in the context of surface design involves a considerable amount of purely arithmetic number manipulation, such as in multiplying matrices or generating displays. The use of this proposed data structure and the overhead of extracting information from it appear likely not to degrade the performance of the system significantly, and, in immensely increasing its "cognitive" powers, would be of great benefit.

REFERENCES

1. Baker, H. F., Principles of Geometry, 6 vols., Cambridge University Press, Cambridge (1922 +).

2. Birkhoff, G., and MacLane, S., A Survey of Modern Algebra, Macmillan, New York (1965).

3. Black, F. S., A Deductive Question-Answering System, (unpublished Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, Cambridge, Mass., 1964).

4. Blatt, H., "Conic Display Generator Using Multiplying Digital/Analog Decoders," Presented at the Fall Joint Computer Conference, Anaheim, California (1967).

5. Cohen, D., and Lee, T. M. P., "Fast Drawing of Curves for Computer Display," Proceedings of the 1969 Spring Joint Computer Conference.

6. Coons, S. A., and Herzog, B., "Surfaces for Computer-Aided Aircraft Design," (Presented at A.I.A.A. 4th Annual Meeting and Technical Display, Anaheim, California, October 1967), American Inst. Aeronautics and Astronautics, New York, no. 67-895, 8 pp.

7. Coons, S. A., Surfaces for Computer-Aided Design of Space Forms, Project MAC Report, MAC-TR-41, Massachusetts Institute of Technology (June, 1967).

8. Feldman, J. A., Aspects of Associative Processing, M. I. T. Lincoln Laboratory Technical Note 1965-13, April 21, 1965.

9. Gleason, A. M., Fundamentals of Abstract Analysis, Addison-Wesley, Reading, Mass., 1966.

10. Johnson, T. E., Analog Generator for Real-Time Display of Curves, M. I. T. Lincoln Laboratory Technical Report, T. T. 398 (July 28, 1965).

11. Jordan, C., Calculus of Finite Differences, Chelsea Publishing Company, New York, N. Y., 1947.

12. Knuth, D. E., The Art of Computer Programming, vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, Mass., 1968.

13. Lee, T. M. P., "A Class of Surfaces for Computer Display," Proceedings of the 1969 Spring Joint Computer Conference.

14. Maxwell, E. A., The Methods of Plane Projective Geometry Based on the Use of General Homogeneous Coordinates, Cambridge University Press, Cambridge (1946).

15. Maxwell, E. A., General Homogeneous Coordinates in Space of Three Dimensions, Cambridge University Press, Cambridge (1961).

16. Roberts, L. G., "Conic Display Generator Using Multiplying Digital-Analog Converters," IEEE Transactions on Electronic Computers, Volume EC-16, Number 3 (June, 1967).

17. Roberts, L. B., "Homogeneous Matrix Representation and Manipulation of N-Dimensional Constructs," The Computer Display Review, Adams Associates (May, 1965).

18. Rovner, P. D., and Feldman, J. A., The LEAP Language and Data Structure, M. I. T. Lincoln Laboratory Document DS-6184.

19. Sproull, R. F., and Sutherland, I. E., "A Clipping Divider," Proceedings of the 1968 Fall Joint Computer Conference.

20. Stotz, R. H., Specialized Computer Equipment for Generation and Display of Three-Dimensional Curvilinear Figures, M. I. T. Electronic Systems Laboratory, ESL-TM-167 (March, 1963).

21. Sutherland, I. E., "A Head-Mounted Three-Dimensional Display," Proceedings of the 1968 Fall Joint Computer Conference.

22. Sutherland, I. E. et al, Computer Graphics Technology, course notes for Applied Mathematics 252r, Harvard University, Fall 1967.

23. Warnock, J. E., A Hidden Line Algorithm for Halftone Picture Representation, University of Utah Technical Report 4-5 (May 1968).

24. Winger, R. M., An Introduction to Projective Geometry, D. C. Heath and Company, Boston (1923).

25. Woods, W. A., Semantics for a Question-Answering System, Report No. NSF-19, Aiken Computation Laboratory, Harvard University, Cambridge, Mass. (1967).

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Harvard University Cambridge, Massachusetts	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP N/A
3. REPORT TITLE THREE-DIMENSIONAL CURVES AND SURFACES FOR RAPID COMPUTER DISPLAY	
4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)* None	
5. AUTHOR(S) *(First name, middle initial, last name)* Theodore M. P. Lee	
6. REPORT DATE 30 April 1969	7a. TOTAL NO. OF PAGES 121 / 7b. NO. OF REFS 25
8a. CONTRACT OR GRANT NO. F19628-68-C-0379 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-69-189 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.	
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Directorate of Planning and Technology, Electronic Systems Division, AFSC, USAF, L G Hanscom Field, Bedford, Mass. 01730

13. ABSTRACT

Rational parameteric polynomial functions of second degree or higher provide a class of curves including all conic sections. They can be generated by an iterative process easily implemented in software or hardware. The numerical accuracy of the process is analyzed. Algorithms for the specification, display, and modification of the curve are presented. Such curves are represented in a homogeneous coordinate formulation convenient for computer applications. Three-dimensional surfaces composed of such curves are similarly convenient to use. Without recourse to trigonometric functions, such classical surfaces as spheres and toroids can be readily described. The ease with which translation, rotation and projective transformations can be applied is exhibited. In particular, we do not perform such transformations on the points of the surface to be displayed -- upwards of several thousand -- but rather upon the rather small set of numbers in a 4 X 4 X 4 tensor that represents the surface. These surfaces are intended to be used in an interactive, freeform computer-aided design system. In this direction we discuss the enforcing of continuity conditions and possible data structures for representing the surfaces.

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
homogeneous coordinates rational parametric cubics interactive computer graphics perspective displays three-dimensional parametric curves computer generated surfaces iterative computation finite-difference methods computer-aided design data structures						